MASTER IN COMPUTER SCIENCE

Game theory based web services collaborative mechanism

Clacens, Kathleen; Goffart, Christophe

Award date:
2011

Awarding institution:
University of Namur

Facultés Universitaires Notre-Dame de la Paix
Faculté d'Informatique
Année académique 2010 - 2011

Game theory based web services collaborative mechanism

Clacens Kathleen - Goffart Christophe

Master thesis submitted in partial fulfillment of the requirements for the degree of Master in Computer Sciences.

Acknowledgement

At first, we would like to thank all teachers and colleagues that allowed us to acquire the knowledge and abilities required for this work that we gathered during our five-year cursus at the University of Namur.

We would like to thank our supervisors Philippe Thiran (Facultés Universitaires Notre Dame De la Paix - Namur) and Jamal Bentahar (Concordia University - Montreal), as well as Babak Khosravifar for their precious advice and support while writing this thesis.

We would also like to thank Magali Piette for the proofreading of our work, although we take full responsibility for potential subsisting spelling mistakes.

Then, we would like to thank our family and friends for the support they gave us.

To conclude, we would like to thank our readers and thesis jury for the interest they manifest.

Abstract

The reputation of communities of web services mainly depends on their web services' reputation. Masters of these communities have responsibility to choose which web services will join them, they consequently have to check their reputation. They cannot ask for this information directly to web services but they can request it from third parties called information services.
Communities pay information services to get information, but web services with a bad reputation can also pay them to lie about their reputation in order to be accepted in communities. In such cases, information services have two choices : on one hand they can tell the truth and not receive reward from web services, on the other hand they can lie and loose the trust communities had placed in them. By means of a decision analysis using game theory and a simulation tool, we look for a mechanism ensuring information services honesty.

Key words: web services, web services communities, reputation, trust, information services, game theory

Résumé

La réputation des communautés de services web dépend entre autres de la réputation des services web qui les composent. Les maîtres de ces communautés ayant la responsabilité de choisir quels services web vont en faire partie, ils se doivent de connaître leur réputation. Ils ne peuvent directement demander cette information aux services web mais peuvent se renseigner auprès d'organismes tiers nommés services d'information.
Les communautés de services web paient les services d'information pour obtenir ces données. De leur côté, les services web ayant une mauvaise réputation peuvent les payer pour qu'ils mentent sur cette dernière dans le but d'être acceptés par les communautés. Les services d'information ont dans de tels cas deux choix : dire la vérité et ne pas recevoir de rémunération de la part des services web, ou mentir et perdre la confiance des communautés.
Au moyen d'une analyse de décision utilisant la théorie des jeux et d'un outil de simulation, nous recherchons un mécanisme permettant d'assurer l'honnêteté des services d'information.

Mots-clés : services web communautés de services web, réputation, confiance, services d'informations, théorie des jeux

Contents

List of Figures

List of Tables

1 Introduction

As time goes on, information technology becomes more and more present in everyday life. As a matter of fact, the trend has been for developers to create always more autonomous systems. The goal of this vision is for users and developers to create, use and share programs and applications in an easy and open way. Types of modeling or implementation are no more obstacles for the use of independent systems.

As most of computer users and developers use the Internet on a daily basis, it becomes an important medium to share this kind of systems. The development of services available on the Internet has therefore been massively studied in the past years. Such services, called *web services*, as defined by the W3C, are *"software system designed to support interoperable machine-to-machine interaction over a network"* [Gro04].

A web service is developed with a particular functionality. For example, a web service functionality could be the search of a currency conversion rate, as well as the booking of a flight ticket. Users address requests to web services that provide the functionality that fulfil their needs. The purpose of web services is thus to answer those requests. The nature of this answer corresponds to the functional aspects of the web service (*What service does the web service deliver ?*). Besides, the way this answer is delivered corresponds to the non-functional aspects of the web service (*How does the web service deliver the service ?*).

Non-functional aspects of a web service are multiple. The speed at which the web service provides an answer, as well as the maximum number of requests the web service is able to handle simultaneously belongs to those non-functional aspects. Other characteristics such as scalability, robustness, accuracy, availability or security can also be listed among those.
Several web services can have the same functional aspect and therefore are able

to answer the same request. However, those web services can differ from each other in terms of their *quality of service*. This quality of service is defined by the non-functional aspects of the web services [Gro11].

When users send requests to a web service, they can send feedbacks about its quality of service. These data are collected in order to measure user's satisfaction. A general opinion about the service is computed using the aggregation of those feedbacks [MS02]. This general opinion is usually known as the *reputation* of the web service.

If a user is satisfied with the results of the interaction with a web service, he will provide a positive feedback. If a single web service receives many positive feedbacks, its reputation is likely to be high. A web service with a high reputation is thus more attractive to new users, as these are assured to get a good service. Therefore, the web service that has a good reputation receives more requests.

As said previously, different web services can provide the *same service*, namely have the same functional aspects. To regulate the amount of requests they receive, web services offering the same service can want to form part of what is called a *community of web services* [EMY+08]. By doing so, requests are dispatched among the several web services members of the community, in order that each web service handles requests without being overloaded.

The dispatch of requests is one of the reasons which explain why it is useful to take part in a community. Indeed, a web service receiving not enough requests can hope to get more, and a web service receiving too much requests can reallocate them.
Communities can also take advantage of collaborating with web services. A heavily requested community can for example attract new web services in order to be sure to handle all the requests properly.

Having web services with good reputation as members can also be beneficial for the community, and therefore be another reason for the latter to attract these web services. Indeed, the *reputation of a community of web services* can rely on its ability to assign a request to one of the web services of the community in a short period a time and on the interest that users have towards this community rather than an other. Nevertheless, the reputation of a community of web services also depends on the reputation of web services which take part of the community. Indeed, the feedbacks about the satisfaction of users has not an impact on the web service alone but on the entire community. [YMB+08].

Because the reputation of a community relies not only on its own management but also on the quality of the web services it contains, a community prefers to

add web services that have a good reputation, as they are more likely to satisfy users. Therefore, when attracting new web services to the community, or when a web service asks for joining, the community needs to check the reputation of the potential future web service of the community in order to decide whether or not the reputation of the web service is satisfactory, namely, whether it suits the expectations of the community. After the checking of the reputation of the web service, the community can decide whether or not the new web service will be accepted.

To make a decision, the community has to find information about the reputation of the web service. Asking directly the information to the web service can put at stake the objectivity of the answer. This problem could be solved by asking the reputation to a third entity. The problem that occurs is that these entities can also be dishonest. An information entity is rewarded by a community to give information but it can also be rewarded by web services to lie to communities. A bad web service can indeed give rewards to an information service if it informs communities that the web service has an higher reputation than it actually has.

The goal of communities is to attract web services with a good reputation and reject web services whose reputations is not satisfactory by sorting out true and false information. For this reason, they need to find a mechanism where trustworthy information entities are used in priority and rewarded, and where information entities giving fake information are avoided and punished. The goal of this method is to ensure the truthfulness of information services and help the communities to use only web services with satisfactory reputation.

The object of our thesis is to study this method and show the results of this investigation.

1.1 Problem and motivation

This thesis focuses on a main concept : web services. A web service is an application that can be found and runs on the Internet and which offers a particular service to users. The main purpose of a web service is to collect requests sent by some users, treat those requests and return adequate responses to the users. They can be used alone or within a composition of web services. Web services can therefore be used to manage "simple" requests but also more "complex" ones if they are in a composition where a heavy task is divided in several "simple" ones among the different web services.

A web service can be defined according to two different aspects.
The first one is the functional aspect, namely the role of the web service. This functional aspect represents the service provided by the web service to the users. It can be really different from one web service to another : calculating the conversion rate between two currencies, foreseeing weather forecast, applying a mathematical function to a data and so on.
The second aspect is the non-functional one. This aspect describes the way the web service manages the requests of the users and can be characterised by different parameters. We can list among those parameters, for example, the capacity of the web service (*How much requests can the web service handle at the same time ?*), its scalability, its robustness, its speed to answer a request or the accuracy of the answer it provides to the user.
Those different parameters put together determine the ability of the web service to provide a service which will satisfy the users. The concept of quality of service of a web service represents this ability [Gro11].

When users interact with web services, they can assess the quality of service of those web services. According to the quality of the service they are provided with, users can be satisfied or not with the web service. The satisfaction (or dissatisfaction) of users can be expressed through feedbacks which are used to build the reputation of that web service. The reputation of a web service is a general opinion about the latter, it is computed through the aggregation of feedbacks provided by users who previously had interaction with the web service [MS03]. If a web service has a good reputation, it indicates that this web service provides good quality service to users. Obviously, a web service with a great reputation receives more requests than a service with a low one, as users want to choose reliable services.

As other service providers (online and offline), users reward web services to handle their requests. The principal goal of web services is therefore to attract users in order to handle their requests and receive rewards in exchange. Indeed, if a web service does not get any request, it can lead to its death. One of the

most difficult dilemmas that a web service has to face is to find a good balance in terms of requests it can handle in order to assure its survival. Unsurprisingly, a web service which is not able to attract enough users, and therefore, requests, will most certainly die in the short run. On the contrary, if a web service is too reputable for its capacity of treatment, namely if it receives more queries that it can manage, it will be overloaded by this high number of incoming requests and will not handle them properly. As a result, users that send these requests provide bad feedbacks that affect the reputation of the web service. If the reputation decreases, no new users will want to address their requests to this web service and, in extreme cases, the overloading can lead to the death of the web service.

To prevent those situations from happening, one solution consists in gathering together web services that have the same functionality. In that configuration, the number of treatable requests of every single service is summed, resulting in a bigger total capacity. A web service that could not attract enough users on its own can benefit from the fact that the requests are now addressed to the group and redistributed among all the members of the group. Therefore, this web service treats more requests than if it was alone. On the contrary, a web service that was previously overloaded by too many requests can now profit from the extended capacity of treatment to transfer some of the extra requests to other web services of the group.

Such a group of web services is called a *community of web services* [EMY+08]. As web services are not on their own anymore but have now to work together as a group, some mechanisms need to be set up in order to achieve coordination between the different entities of the community. A web service is chosen among all the services of the group to lead the community. This special web service is called the *master of the community* while the others are its slaves.
The master of a community has several specific tasks [EMY+08]. It is the intermediary between the community, the web services composing the community and the users. Indeed, the master receives requests addressed to the community and then dispatches them to its slaves. When the response to a request is ready, the master is in charge of sending it to the corresponding user.

As web services, communities of web services also have their own reputation. The reputation of a community of web service depends on its own capacity to satisfy users' needs. In one hand, this satisfaction can be measured with parameters such as the speed at which the master assigns a request to a slave or the preference of users to use this community of web services before an other one for example. On the other hand, the satisfaction of a user also depends on the quality of service of the slave web service that handles its request. Therefore, it is obvious that if communities want to have good reputation, they need to be composed by good reputation web services [YMB+08].

Apart from the reception and sending of requests, the master is also responsible for deciding whether or not a web service can join its community. When a web service asks for joining a community, the master has to check if the reputation of the web service corresponds to the expectations of the community. Web services are motivated to enter a community of web services because it can help them survive by regulating their incoming requests. Communities also have motivations to attract new web services. Indeed, if the total number of requests addressed to the community is too low, the master of the community can attract new web services that are reputed to have lots of requests. On the contrary, if there are too much requests coming to the community, the master can attract new web services in order to have a bigger number of web services to distribute requests. A community can also want to attract new web services in order to profit from their reputation.
The master can attract new web services, decide whether or not a web service can enter the community and even decide to eject a certain service from its community if it considers that this web service does not reach the community's expectations anymore. The master must also try to retain the members to leave the community.

In our study, we take a closer look to this issue and analyse the case of a master having to decide if a new web service should be accepted or not in the community. Web services with a bad reputation do not attract many requests. Therefore, these web services can want to enter a community of web services in order to get more requests. However, if the web service has a bad reputation, it is not likely that community masters will accept them. There is a risk that these bad web services try to pretend to have a good reputation when asking to join a community. The master, unable to verify correctly that assumption, could be fooled and decide to accept the web service in its community. He would only realise after a while, when requests have been sent to this web service, that it was a bad decision and eject the bad service out of the community. Meanwhile, the bad impact of the web service on the community has been done as bad feedbacks from users whose request have been handled by the bad web service affect the reputation of the community. Therefore, it is crucial for masters of community to make the right decision when accepting or rejecting a web service wanting to join, but, it order to do so, masters need to have the right information about web services.

The decision to accept a web service into a community or not is complicated due to the lack of valuable information for masters. The goal of our thesis is to find a way to ensure that masters of communities obtain such trustworthy data. We decide to use a third party called *information service* to solve this issue.
Information services are the intermediary between web services and communi-

ties. Information services collect feedbacks from web services users and, based on these feedbacks, constitutes a repository of the reputation of all web services. When the master of a community receives a request from a web service to join the community, it has the possibility to ask information services for information about this particular web service. In order to get the most accurate information, a community master does not address to one but to several information services and can thus compute an average reputation value based on the reputations announced by those information services. Depending on this average reputation, the master is thus more able to decide safely whether or not it should allow the web service to enter the community.

If all information services are honest entities, there is no decision issue. The only fact of asking about a web service to an information service would assure masters of community to have the right information and make the right decision according to the latter.
The problem that occurs in this 3-actors architecture (web service - community - information service) is that communities can reward information services for the information they get, but web services can also be tempted to reward information services in order to force them to lie. A web service with a bad reputation is not going to be accepted by any community if it does not reach their expectations. In order to enter a community, the bad web service can offer to the information service a reward if, when informing communities, the information service pretends that the web service has an higher reputation than it actually has.

The final motivation of information services is to get as much rewards as possible. In this architecture, information services can get their rewards from two different sources : communities reward them to get information and web services can reward them to lie. However, these two sources of reward are not compatible. Indeed, if information services accept to lie in favour of the web service, the community that was fooled will no more have trust towards those information services. As the community realises that the quality of the web service does not match the one announced by the information services, the community will not be eager to request more information from information service that provided incorrect or even fake information.
If an information service always lies, the trust that communities have toward it gradually decreases until no community asks for its information. If such situation is reached, the information service is no more rewarded. Indeed, if no community asks for information from this information service, web services have no more motivation to reward the information service, as the latter will have no more possibility to lie.
An information service must decide wisely when it must lie and then get rewarded by web service, and when it is better to tell the truth, in order to get rewarded by communities for the right information and keep their trust to assure future

interactions.

Information services have to choose, each time they are asked for information, between telling the truth and lie about the reputation value they know. It is a complicated decision as information services want to maximise their profit and to keep trustful relationships with communities. Furthermore, an information service never works alone as an information source. Indeed, master of communities asks several information services at the same time in order to get the best approximation of the actual wanted reputation value. In that situation, as other's actions impact its own, no information service can elude them. An information service, before making a decision, has to calculate what will be the consequences of its choice, depending on other one's. The best action to make would be the one that provides the bigger reward no matter what other information services' actions are.

Situations with entities interacting in such a way, where one's decision impacts other's profits, are exactly what *game theory* studies. In that theory, rules can be applied in order to determine, for each interacting entities (called *players*), which action is the best to take in order to reach a particular goal (one player's biggest gain, biggest total gain, ...).

The game theory is perfect to model information services situations and choices. Based on this, we are able to study in which situations information services lie, and in which cases they prefer to tell the truth.

The goal of our study is to find which elements must have a web service network in order to force information services to tell the truth. The final aim would be to have a *trustful network* in which information services always give true information and masters of communities of web services only accept in the community web services that correspond to the community level and expectations.

1.2 Methodology

Web services are the key elements of our thesis. In Chapter 2, we study the main concepts related to the latter. We also present the idea of communities of web services and develop the notions of reputation and trust.

Our aim is to find a mechanism to ensure honesty among information services. As communities of web services always ask several information services at the same time, the outcome of a single request depends on each of these information services' answer. When an information service chooses between telling the truth or lying, it must therefore take other information services' actions into account. Such a situation can be modelled as a game. We therefore introduce in Chapter 3 the basics of the game theory.

In Chapter 4, we define in a specific and detailed way the problem we study and develop the theoretical solution we propose.

In order to assess the validity of our solution, we implemented a simulation tool that we present in Chapter 5. A complete and detailed presentation of the realisation of this simulator can be found in Appendix A.

In Chapter 6, we show the results we obtained with the use of our simulation tool. We set up a number of test environments that help us to study the most interesting cases. Then we compare the outputs computed by the simulator. These comparisons allow us to draw some conclusions about the different constituent elements of these environments.

We present the conclusion of our investigation in Chapter 7.

Finally, some of the limitations of our work are presented in Chapter 8 as well as some suggestions for potential future works.

2 Preliminaries

In this section, we deal with some of the main concepts necessary for the comprehension and development of our problem and solution.

First, we present some notions about web services and communities of web services.

Secondly, we define two important concepts : the reputation and the trust.

Contents

2.1 Web services and communities

2.1.1 Web service

As time goes by and technology evolves, Internet becomes more and more present in our everyday life as many of our daily activities requires its use. This evolution gave birth to a new kind of market, and a new way of shopping : *Web services*.

Web services have the same purpose as human services : their goal is to receive users' requests and answer to those the best they can. One of the main reasons for the use of web services is the development of loosely-coupled, cross-entreprise business processes (which are also called B2B applications). This means that web services can be used without having to worry about how it is implemented or where in the world it is located.

The W3C defines (in [Gro04]) a web service as follows :

Definition 1. *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*

The use of a web service results of the interaction of three different actors, as shown in Figure 2.1.

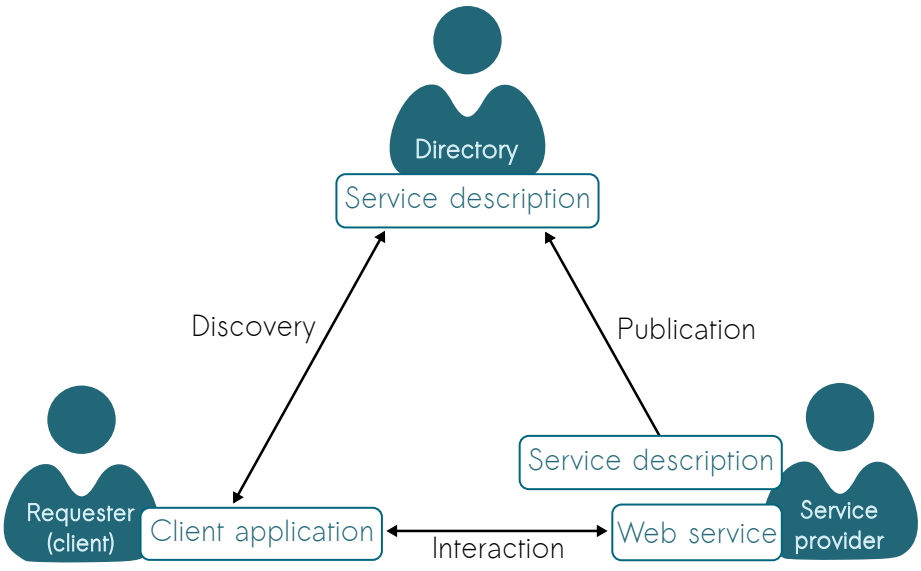

Figure 2.1: Web services' usage actors

The first main actor is the *Service provider*. It is responsible for the creation of the web service and performs the handling of requests by computing an appropriate answer. The service provider submits a description of the operations the web service can offer. This service description is then published on an online directory.

The *Directory* serves as discovery agency where the service's descriptions are stored and made available to the users. When a user needs a particular service, it browses through the directory in order to find the most suitable web service according to the description establish by the provider of the latter.

When the right web service has been found, the user can begin the interaction with the service by sending requests. The user can also be called the *Requester* and acts as the third actor.

The use of a web service leads to the implication of several protocols. As shown in Figure 2.2, a first protocol is used by the service provider and the directory to describe the web service. This protocol is *WSDL* (which stands for *Web Services Description Language*). It consists of a XML-based language specially designed for that purpose and standardised by the W3C [CCMW10].

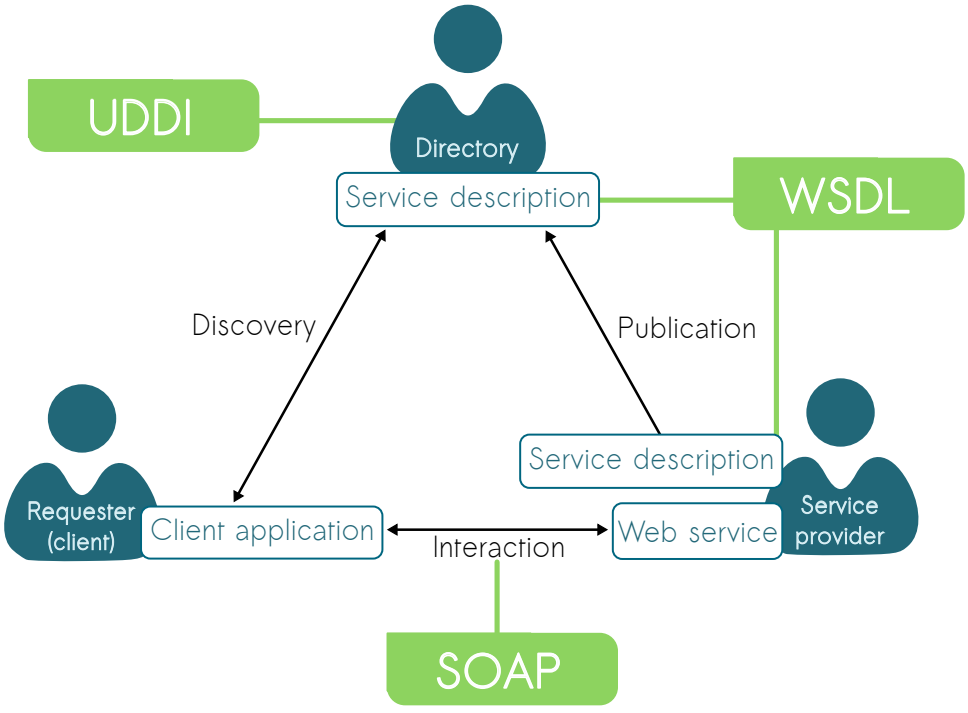

Figure 2.2: Web services' usage protocols

The second used protocol is *UDDI* and stands for *Universal Description, Dis-*

covery and Integration. UDDI allows service providers to publish the description of their services on the directory but also the users to discover the web services. The UDDI protocol has been standardised by OASIS [udd10].

If we now consider the way different web services interact together, it is important to find a common message format so that web services can understand each other and share information. We can achieve that by using *SOAP* (for *Simple Object Access Protocol*). SOAP is *"a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment"* [GHM+10]. As for WSDL, SOAP is standardised by the W3C.

2.1.2 Community of web services

Web services can have similar functionalities. For example, one can find two web services that offer currency converting operations. In this case, it is possible to gather the web services into groups that we refer to as *Communities of web services*. In a community, we find two types of web services : a unique *master* and its *slaves*. The master takes the responsibility of leading the community and has three main tasks. [EMY+08]

The first task consists of the management of the community. The master *"monitors all events happening in a community such as the arrival of new Web services, departure of existing ones, identification of Web services to be part of composite Web services, and imposing sanctions on Web services in case of misbehaviour."* [EMY+08]
The master can also decide to dismantle the community if it finds that the number of web services in the community falls below a certain threshold or if *"the number of participation requests in composite Web services that arrive from users over a certain period of time is less than another threshold"* [EMY+08].

The master of a community is also responsible for attracting new web services into the community and retaining the ones that are already in the community. This task is important because, as said in the previous paragraph, the community could be dismantled if the number of web services that are members of the community is too low. In order to avoid such a situation, the master consults the directories to discover new web services. When the master finds a web service with functionality compatible with those of the community, the provider of the service is contacted.

Finally, the third task of the master consists of the nomination of web services in order to participate in composition scenarios.

At this point, the questions of why a web service would try to join a community and why a community would want to attract new web services arise. If we consider a single web service, the problem is related to the number of requests the web service has to handle. Indeed, a web service has a limited capacity of treatment regarding the incoming requests and will try to avoid two situations. On one hand, the number of requests could be too low compared to its total capacity. In this case, by joining a community, the web service would receive more requests as the requests sent to the community would be distributed among all its members. On the other hand, the web service could be overloaded by a number of requests exceeding its capacity of treatment. By entering a community, the web service would avoid the overloading as the exceeding requests would be distributed to other members of the community.

As for the community, we learned that the master could decide to dismantle its community if the number of web services inside is too low. In order to avoid this situation, the community tries to attract new web services.

2.2 Reputation and trust

2.2.1 Reputation of web services

As for non-Internet based services, like buying a flight ticket in a travel centre, it is possible that several web services have the same functionalities. We therefore need a notion that helps us to differentiate those different web services. We use the concept of *reputation*. It is considered in the exact same way with web services as with non-internet based services : to go on with the example introduced in the beginning of this extract, it is more likely that someone wanting to buy flight tickets will choose a more reputable travel centre than a less reputable one.

We can describe the reputation as the opinion of the public towards a person, a group of people or an organisation [MS02]. The problem of this definition is that we cannot base our concept of reputation only on the opinion of several users. User ratings indeed tend to be subjective and can also be easily manipulated. As a result, we need to use more objective metrics to compute the reputation of a web service. We will here consider the *Quality* and the *Market Share* [KBMT10].

The Quality represents the capability of a web service to handle the users' requests in a timely fashion. It is found by collecting all the rates given by the users to this web service. We then compute the ratio between the set of positive feedbacks and the set of all the feedbacks. In order to deal with selfish agents that dynamically change their behaviours, we give more importance to the recent

information by using a timely relevance function.

The Market Share indicates the extent to which the web service is active in the network of the providers of services.

2.2.2 Reputation of communities of web services

We just saw in Section 2.2.1 how to find the reputation for a single web service. However, in the case of a community, the metrics we use are not the same. Moreover, the reputation of a community can be computed according to two perspectives : the one of a user or the one of provider of web service. Here, we only consider the user's point of view. The reputation is thus based on the three following metrics : the *responsiveness*, the *inDemand* and the *satisfaction*.

When a user sends a request to a community, the master has to nominate a slave from his community that can handle that request as fast as it can. The faster the master manages to do this task, the better it is. This speed is represented by the responsiveness metric. The inDemand shows the interest of the users towards a certain community against the others. The satisfaction simply corresponds to the subjective opinion of the clients. We can also note that the responsiveness and the satisfaction are direct evaluations of the interactions between the users and the community of web services while the inDemand is an assessment of a community in relation to the other communities.

Another useful concept is the notion of *performance*. In order to obtain it, we need a performance function measuring the aptitude of a community to act successfully, that is to say having a optimal use of the allocated resources and an optimal market share. The three needed elements to compute the performance of a community are the reputation, the inDemand and the capacity of the community. The capacity corresponds to the maximum number of requests that a community can handle. An acceptable performance is obtained if a community has a good reputation and an optimal balance between its inDemand and its capacity.

The reputation can explain why it could be interesting for a single web service to join a community. A web service alone can easily be overloaded by an intense flow of users' requests. It will lead to a poor responsiveness and therefore a fall in the users' satisfaction. As a result, the web service will see a drop in its reputation. A solution is to group different web services in a community. By aggregating the total number of requests that each single web service can handle, and redistributing them equitably among all its members, a community will be granted a better availability and hence better performance. On the contrary, it is

possible that a single web service is not able to attract enough users in comparison with its capacity. In this case, joining a community will help the web service to obtain more requests to handle.

Having a good reputation can be double-edged. It will indeed bring to the reputable web service a large amount of requests. The consequence of that demand can be an overloading. The challenge here is to find a tradeoff between one's capacity and market share so that a web service is neither overloaded nor idle. Once such a tradeoff has been found, a web service should see a stabilisation of its reputation and its market share level. [KBM+10]

Now, the arising question of why a web service would be encouraged to join a community with a lower reputation that can possibly degrade its own reputation finds its answer quite easily. Survival is simply a more critical goal than carrying on the current level of reputation. Therefore, joining a community is encouraged for a web service even if it decreases its reputation as long as the total performance level is higher.

2.2.3 Reputation-based architecture

In order to work with the concept of reputation, a system needs to have an architecture using the two following specific elements : some registries containing entries that describe individual web services and a reputation system [EMY+08]. The registries can be implemented using the UDDI protocol which defines a standard method for publishing and discovering the network-based software components of a service-oriented architecture [udd10]. The reputation system is the core component of a reputation-based architecture and has two main functions. The first is to maintain a repository of run-time operational data that are needed to compute the performance metrics of a community and the second is to rank the communities according to their reputation using a ranking algorithm. The reputation system is also referred to as the *Controller Agent* and is also responsible for detecting malicious acts in the system.

User and provider agents, proxies between respectively the user and the provider and the other parties, are also needed. It is important that they remain independent in order to intercept trusted run-time data about each web service interaction.

2.2.4 Trust

The advantage of using service-oriented architectures is that we can now create applications that interoperate in a smooth way and whose components are loosely

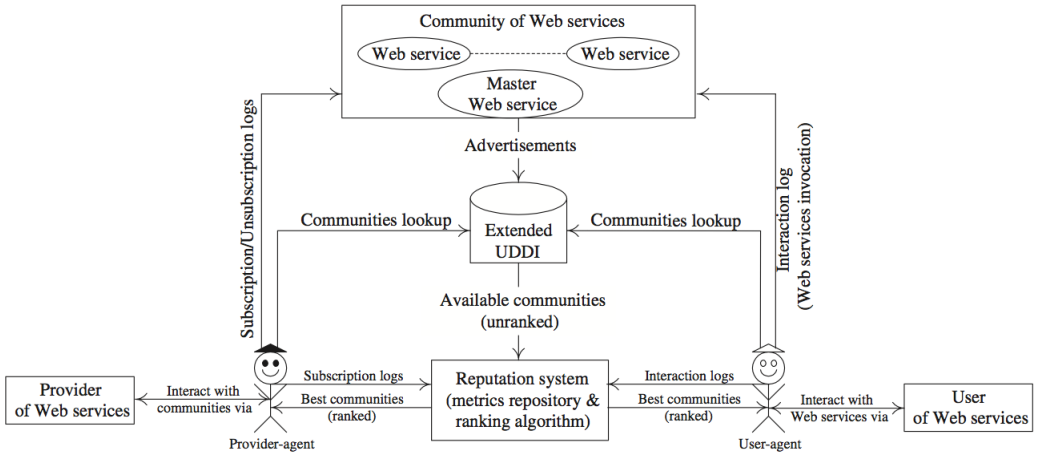

Figure 2.3: Architecture of reputation-based Web services communities

coupled. The question that arises is how to select services for the purpose of composition scenarii. An important criterion for this selection of services can be the *trust*. Trust can be defined as *"the measure of willingness that the trustee will fulfil what he agrees to do"* [BKG09]. A *"new trust model for service selection based upon social network analysis to capture the emergence of trust via service networks"* was proposed in [BKG09].

In this model, we can distinguish two sets of services : the *customers* and the *providers*. All of these services belong to the same community in the sense that a service can share information with the other services through the network.

Assessing the trust of a provider can be done using two types of assessments. The first is called *direct assessment*. It can be computed if the customer has enough transactions with the provider. However, it is likely that sometimes a customer did not have a sufficient number of interactions with a provider. In this case, the customer uses the network to get information about the provider from other services that know this provider and are willing to share their knowledge. This technique is called *indirect assessment*.

When a customer asks another customer (that we will refer to as informant) for information, the latter has two possible strategies : he can tell the truth or lie. Obviously, we want to reach a situation corresponding to a Nash equilibrium in which informants have no better choice than telling the truth. This can be achieved through a *3-step incentive mechanism* where informants obtain rewards or penalties according to their strategies in 3 steps.

The strategy x_k of each agent c_k is defined in terms of the provider's trust

value this agent reveals. Thus, the utility function of a customer agent c_k is defined as follows :

$$u_k(x) = f_k(x) + g_k(x) + c.h_k(x)$$

where

- $f_k(x) > 0$
- $f_k(x) < |g_k(x)|$
- $f_k(x) + |g_k(x)| < |h_k(x)|.f_k(x)$.

The *first step incentive* $f_k(x)$ is a positive reward that a customer gives to an informant willing to give her information. The *second step incentive* $g_k(x)$ corresponds to a value that will be granted to a informant depending on the similarity between the information it gave and the average information revealed by the other informants. This second value can be negative if the difference is high and therefore acts as a punishment preventing informant to reveal incorrect information. Finally, after having used the provider, the customer can tell whether it fits the informants' predictions about the provider's behaviour. The difference between what was expected and what was actually experienced is used to calculate the *third step incentive* $h_k(x)$. Of course, the latter can only be considered if the customer decides to have a transaction with the provider. In this case, c will be set to 1 in the utility function.

It is important that each incentive has to be higher than the sum of the previous ones, that is to say $f_k(x) < |g_k(x)|$ and $f_k(x) + |g_k(x)| < |h_k(x)|$. It guarantees the *incentive compatibility* property which means that each informant will reveal exactly what they believe about providers or, in other words, they will tell the truth. This can be proved by using the three following lemmas :

- The first says that revealing the true trust value about the provider is a Nash equilibrium strategy in the trust game.
- The second lemma affirms that if the provider is untrustworthy, revealing a false trust value about it is not a Nash equilibrium strategy in the trust game.
- On the other hand, if the provider is trustworthy, revealing a false trust value about it is a Nash equilibrium strategy in the trust game. This is the third lemma.

From this three lemmas, we can tell that a customer has an incentive to lie only if the second step incentive $g_k(x)$ is strictly positive and the third step incentive is not obtained ($c = 0$). In this case, the gained utility will be $|f_k(x) + g_k(x)|$. However, by telling the truth, he can get the three incentives and the gained utility will be $|f_k(x) + g_k(x) + h_k(x)|$. And so, telling the truth is the best strategy under Nash equilibrium.

3

Resolution technique : game theory

Information services are agents interacting with both web services and communities of web services. When they are requested to give information about a web service, they can either choose to tell the truth and give the actual information or lie and provide a false one. Each of these actions creates a different profit and affects differently the relationships the information service has towards the two other agents. Furthermore, an information service is never requested alone as community masters always request several information services at the same time. Results provided by the other information services also affect the profit and the relationship between the information service and the community on one hand, and between the information service and the web service information is asked about on the other hand. For these reasons, an information service decision to tell the truth or lie depends on other agents' actions.
The use of game theory can help us to model such situations as games, putting these agents in the roles of players. It will help us to analyse the different options an information service faces, and understand what are the reasons that make a choice seem better than another.

In this Chapter, we begin by making the concept of game more concrete by presenting a well-known example in Section 3.1. We then take a quick overview of this discipline basis concepts in Section 3.2 where we introduce concrete notations. In Section 3.3, we talk about one of the main notions of game theory, which is the Nash equilibrium. To finish, we take a look at the Pareto efficiency in Section 3.4, where we see how we can identify a state where the welfare of each player is taken into account.

This Chapter has been written thanks to the help of the game theory book [OSB09].

Contents

3.1 Theory presentation

Game theory can be applied in many fields where people interact such as politics, economy, social issues or games. This discipline simplifies those everyday life situations in which decision makers have to take actions by modelling them as what is called *games*. Like in chess, depending on how players move their paws, the outcome of the game changes. The analogy is just that simple : if we take business men as players, depending on how they manage their paws (financial resources for example), the outcome of the game (the market situation) changes.

3.1.1 Examples

One night, two thieves break into a little shop along the street. These armed burglars steal all the money. Two days after, the police track and arrest two people separately. Those two suspects, Cameron and Kevin, are held in different rooms at the police station and have no way to communicate.

The police face a problem as they collect clues to accuse both of them of minor offense, but not enough proofs to convict any of the two lads for the major crime of armed robbery.

To obtain confessions, the police offers each of the two people to "make a deal". The deal works as follows :

- If both decide not to talk, they will go to prison for 1 year.
- If one confesses and the other stays silent, the first one is free whereas the other goes to prison for 5 years.
- If both of them confess, they both have a 3 years imprisonment sentence.

Both Cameron and Kevin have to decide on their own which decision they want to take, knowing that the outcome of the situation will depend not only of their own choice, but also on the choice of the other one.

Such situations, where people have to make choices, and where each decision-maker does not master every variable of the problem, and thus depend and interact with other ones, can be modelled as games.

Such games are what game theory studies, and the one described above is one of its most famous examples called "*The prisoner dilemma*".

In order to be easier to use, let's represent the prisoner dilemma as an array :

		Kevin	
		Talk	Not Talk
Cameron	Talk	3,3	0,5
	Not Talk	5,0	1,1

Table 3.1: The prisoner dilemma - representation of possible sentences in terms of years spent in jail

In Table 3.1, we can see more easily what are the outcomes (in terms of years in jail sentence), for each person, according to each choice combination, the first number always being the outcome for Cameron, the second one being the outcome for Kevin.

3.2 Basic concepts

3.2.1 Payoff function

As normal human beings, both lads prefer freedom to jail. If they have to go anyway, the shorter is the better, we all agree on that.

This really natural idea is the base of a mathematical concept called the *payoff function*[1]. This function assigns a value to each of the possible outcomes, increasing the value as the satisfaction regarding an outcome improves. An outcome is preferred to another one only if the value of the payoff function of the first one is higher than the one of the second. Seeing it the other way around, if we have u, the payoff function, and a and b, two different outcomes of a game : $u(a) > u(b)$ if and only if the decision-maker prefers a to b

Such a function is useful in terms of reading and comparison. Indeed, in Table 3.1, the best outcomes can easily be figured out as the context is explained just before. Knowing the figures stand for years in prison, everyone knows that we are looking for the shortest one, but this is not the case for every situation.
In another context, the biggest figure could represent the best outcome. In a third one, it could be impossible to express outcomes as figures.
A payoff function solves this kind of communication and comprehension problems : the higher the better, and equality represents indifference.

Let's go back to our two suspects and apply to them the concept of the payoff function. The worst that can happen to any of them is to have to go to prison for 5 years, we can assign to such an outcome a payoff value of 0. Staying in jail

[1]Other names for this function are *preference indicator function.* or *utility function*

for 3 years is a little better, and for only 1 year is much better, the payoff values are respectively of 1 and 2. The best outcome is to be free, and deserves a payoff value of 3.

		Kevin	
		Talk	Not Talk
Cameron	Talk	1,1	3,0
	Not Talk	0,3	2,2

Table 3.2: The prisoner dilemma - representation of possible sentences in terms of payoff values

The preference between outcomes are said to be "ordinal", that is to say a payoff function can help us figure that a player prefers an action a to an action b, which payoff value is a little bigger, and prefers the action b to a third action c, which payoff value is even bigger, but it cannot define "how much" an action is preferred to one other.

3.2.2 Game analogy

We use the game concept to model interactions between decision-makers and extend the analogy to all the aspects of these interactions. We refer to what we called until now decision-makers as *players*. Those players have to make a choice between different *actions*, and $A(i)$ is the set of all the actions $a(i)$ available to player i. We talk about an *action profile* when we want to refer to one of the situations, where each player chooses one of its actions.

As we saw before, some outcomes are better than others, so players have preferences about the action profiles.

3.2.3 Strategic game with ordinal preferences

The first model of game theory we consider is the model of *strategic game with ordinal preferences*. We use the qualification "with ordinal preferences" to insist on the fact that when an action is chosen, it will be chosen every time the game is played.

This concept is defined as follows [OSB09] :

Definition 2. *A strategic game (with ordinal preferences) consists of :*

- *a set of players*
- *for each player*
  - *a set of actions*
  - *preferences over the set of action profiles*

3.2.4 Best response function

Let's go back to our prisoner dilemma, and let's try to analyse what can happen.

- Both suspects will do everything they can to benefit from the best possible outcome.
- The best outcome would be freedom (payoff value of 3), so both of them are interested in *talking*.
- But they both know that *talking* is the action the other will also certainly prefer. And if both of them *talk*, both of them go to jail for 3 years (payoff value of 1).
- At this point, *not talking* can be considered, as it would mean only 1 year in prison (payoff value of 2). But this means having faith in the decision of the other suspect, hoping that she will *not talk either*. Not talking means taking the risk of falling into the worse possible situation, which is letting a traitor partner gain freedom while having to sleep in jail for 5 years (payoff value of 0).

In some situation, suspects have total faith in each other and have no fear of staying silent, in other situation, suspects know their partners are selfish and will always talk, but let's take the situation where one suspect, say Kevin, isn't sure about Cameron's behaviour. One of the best options for him is to compare actions and see what is the best action to take, in each case.

- If Cameron *talks*, then Kevin has the choice between staying 3 of 5 years in prison, the best outcome for him if the first one, so he *talks* too.
- If Cameron *does not talk*, then Kevin has the choice between being free, or staying 1 year in prison, once again he prefers the first outcome and decides to *talk*.

To draw up this list of actions, which are the best to choose according to each of the other players' actions, is in fact building what we call the *best response function*.

The definition of the best response function B_i of a player i is [OSB09] :

Definition 3. $B_i(a_{-i}) = \{a_i \text{ in } A_i : u_i(a_i, a_{-i}) \geq u_i(a'_i, a_{-i}) \text{ for all } a'_i \text{ in } A_i\}$ *where :*

- a_i *is the action taken by the player* i
- a_{-i} *is the list of other players' actions*
- $u_i(a'_i, a_{-i})$ *is the payoff value for the player* i *for the action profile* (a'_i, a_{-i})

In this definition, we can see that the application of this function B_i to an action a of an other player results in the set of all the best actions i can play if the other player plays a. Those best actions are the ones whose payoff value is at least as good as the one of any other action.
If we apply those notations to our example, we get :

- for Cameron
  - $B_{Cameron}(Talk_{Kevin}) = \{Talk\}$
  - $B_{Cameron}(Not\ talk_{Kevin}) = \{Talk\}$
- for Kevin
  - $B_{Kevin}(Talk_{Cameron}) = \{Talk\}$
  - $B_{Kevin}(Not\ talk_{Cameron}) = \{Talk\}$

3.3 Nash Equilibrium

3.3.1 Introducing new examples

To strengthen comprehension of already seen concepts, and to ease illustration of following ones, let's take a look at two other well-known examples.

- **Bach or Stravinsky ?**
  Bernard and Stan are two friends and they would like to go to a concert. They do not have the same tastes : Bernard likes listening to Bach whereas Stan prefers the music of Stravinsky. However, each of them would rather go to a concert with a friend, even if the music is not his favourite, than go to a concert alone.

		Stan	
		Bach	Stravinsky
Bernard	Bach	2,1	0,0
	Stravinsky	0,0	1,2

Table 3.3: Bach or Stravinsky - game representation

This situation is modelled in Table 3.3 :

The best response functions are :

- for Bernard
  - $B_{Bernard}(Bach_{Stan}) = \{Bach\}$
  - $B_{Bernard}(Stravinsky_{Stan}) = \{Stravinsky\}$
- for Stan
  - $B_{Stan}(Bach_{Bernard}) = \{Bach\}$
  - $B_{Stan}(Stravinsky_{Bernard}) = \{Stravinsky\}$

- **Matching pennies**
  Dorothy and Stuart are playing a simple game. Each of them has a coin and can choose to show one side of the coin or the other. At the same moment, they have to show the side they chose to the other player. Dorothy wins the game if the sides are different, but if they are the same, Stuart is the winner

  This game is modelled in Table 3.4:

		Stuart	
		Head	Tail
Dorothy	Head	0,1	1,0
	Tail	1,0	0,1

Table 3.4: Matching pennies - game representation

The best response functions are :

- for Stuart
  - $B_{Stuart}(Head_{Dorothy}) = \{Head\}$
  - $B_{Stuart}(Tail_{Dorothy}) = \{Tail\}$
- for Dorothy

* $B_{Dorothy}(Head_{Stuart}) = \{Tail\}$
* $B_{Dorothy}(Tail_{Stuart}) = \{Head\}$

3.3.2 Domination

If we take a look at the best response function of any of the suspects in the prisoner dilemma, we can observe that, no matter what the other does, the best thing to do is always to talk. An action with this property is called a *dominant action*, and the other ones are called *dominated actions*. Not every game has a dominant action. Indeed, the *prisoner dilemma* is the only one of the three given examples to have such an action.

There are two levels of dominance :

- *Strict domination* appears when an action is better than another one, no matter what the other players' actions are. It can be defined as follows [OSB09] :

  Definition 4. *Player i's action a_i'' strictly dominates her action a_i' if $u_i(a_i'', a_{-i}) > u_i(a_i', a_{-i})$ for every list a_{-i} of the other players' actions*

- *Weak domination* appears when an action is at least as good as another one no matter what the other players' actions are, and better than the other one for some of the actions of the other players. It can be defined as follows [OSB09] :

  Definition 5. *Player i's action a_i'' strictly dominates her action a_i' if $u_i(a_i'', a_{-i}) \geq u_i(a_i', a_{-i})$ for every list a_{-i} of the other players' actions*

3.3.3 Nash equilibrium

Let's get back, once again, to our prisoner dilemma. We saw with the *best response function* that talking was the best thing to do, or at least the best response to give to any action of the other suspect. Let's suppose that both Cameron and Kevin decide to talk, we are thus in the situation where both of them will go to prison for 3 years. According to the set of outcomes, this is the second worst result one can get (payoff value of 1), so not what you would expect someone to look for. Nevertheless, this is the best each suspect can have, if the other one does not change his mind.

When we reach such a point in a game, where no one can improve its situation by changing his action if the other players keep their actions, we face a *Nash Equilibrium*. The concept of Nash Equilibrium is defined as follows [OSB09] :

Definition 6. *The action profile a^* in a strategic game with ordinal preferences is a Nash equilibrium if, for every player i*

$$u_i(a^*) \geq u_i(a_i, a^*_{-i})$$

for every action a_i of player i, where u_i is a payoff function that represents players i's preferences.

A Nash equilibrium corresponds to what we can call a *steady state* : if, whenever the game is played, the action profile is the same Nash equilibrium, then no player has a reason to choose any different action. It can also express a *stable social form* : if everyone adheres to it, then no individual wishes to deviate from it.

3.3.4 Examples' Nash equilibria

Let's take a look to our example situations and determine for each of them their Nash equilibrium :

Prisoner dilemma

		Kevin	
		Talk	Not Talk
Cameron	Talk	1,1	3,0
	Not Talk	0,3	2,2

Table 3.5: The prisoner dilemma - Nash equilibrium

As we saw above : if one of the suspects talks, the other one cannot do anything else but talk if she wants to minimise her sentence.

Bach or Stravinsky ?

		Stan	
		Bach	Stravinsky
Bernard	Bach	2,1	0,0
	Stravinsky	0,0	1,2

Table 3.6: Bach or Stravinsky - Nash equilibria

We can observe that this situation offers two Nash equilibria. Indeed, if one of the two friends decides to go to a concert, whichever it is, the best for the

other one is to go too : even if he is not fond of the composer, he would rather go out with his friend than alone.

Matching pennies

		Stuart	
		Head	Tail
Dorothy	Head	0,1	1,0
	Tail	1,0	0,1

Table 3.7: Matching pennies - no Nash equilibrium

In contrast with Bach or Stravinsky example, where we had two Nash equilibria, in this case, we have *none*. The rules of the matching pennies game are designed is such a way that no matter which situation we are standing in, there is always one of the two players who can do better by changing her action.

Our three little examples are enough to highlight an important observation : in a ordinal game, there is not always a Nash equilibrium, and if there is, there can be more than one.

3.3.5 Strict and non-strict Nash equilibrium

The Nash equilibrium can bring in a payoff value that is better for a player than any other one if the other players keep their actions. If it is the case for every player, if changing an action means having a worse payoff for everybody, then we have a *strict Nash equilibrium.*
But if there is another payoff that is not worse, but only "no better" (which means having a equality), we have a *non-strict Nash Equilibrium.*

3.3.6 Nash equilibrium and best response function

We recall that the best response function of a player is the list of the actions that are the best to choose according to each of the other players' actions (Definition 3). We can link this function to the concept of Nash equilibrium as follows [OSB09] :

Definition 7. *The action profile a^* is a Nash equilibrium of a strategic game with ordinal preferences if and only if every player's action is a best response to the other players' actions.*

In situations where players do not have a lot of action possibilities, finding a Nash equilibrium by examining each action profile is possible, but when the number of actions grows, using the best response functions to find a Nash equilibrium is a better idea.

3.3.7 Nash equilibrium and domination

According to Definition 4, a strictly dominated action will never be the best to choose as there will always be a better one, no matter what the other players do. A strictly dominated action will thus never be used in a Nash equilibrium.

According to Definition 5, a weakly dominated action is only worst for some of the other player's actions, so, in a strict Nash equilibrium, no player's equilibrium action will be weakly dominated, but, such an action can belong to a non-strict Nash equilibrium.

3.4 Pareto efficiency

Another interesting state in games, besides Nash equilibrium, is what is called a *Pareto optimum*. The concept of *Pareto optimality* (or *Pareto efficiency*) appears with an action profile where no one can improve its gains without degrading another one's. When players are in a situation where no player can increase (even a bit) its outcome without decreasing other one's, then players reached Pareto efficiency.
Such action profiles reflect situations where players tend more to maximise the global welfare rather than the outcome of oneself.
As for Nash equilibrium, where we have strict and non-strict equilibrium, a *Pareto optimum* can be weak or strong [OSB09].

Definition 8. *An action profile a^* is a weak Pareto optimum if there is no action profile strictly preferred over a^* by every player. a^* is a strong Pareto optimum if there is no action profile considered at least as good as a^* by every player and strictly preferred by at least one player.*

Let's look for Pareto optimums in our well-known examples :

Prisoner dilemma

We see that in this situation, if no one talks, neither Kevin nor Cameron can change her decision and improve her outcome without making the other one's worse. The Pareto improvement is not the same action profile as the Nash equilibrium, and we can see that both outcomes would be improved by choosing the

		Kevin	
		Talk	Not Talk
Cameron	Talk	1,1	3,0
	Not Talk	0,3	2,2

Table 3.8: The prisoner dilemma - Pareto improvement

first one (from payoff value of 1 to payoff value of 2).

Bach or Stravinsky ?

		Stan	
		Bach	Stravinsky
Bernard	Bach	2,1	0,0
	Stravinsky	0,0	1,2

Table 3.9: Bach or Stravinsky - 2 Pareto improvements

In this example, we can see that Nash equilibrium and Pareto improvement are the same action profiles. Indeed, even if one of the friends is listening to a concert he dislikes, he could not increase his payoff value without decreasing the one of his pall.

Matching pennies

		Stuart	
		Head	Tail
Dorothy	Head	0,1	1,0
	Tail	1,0	0,1

Table 3.10: Matching pennies - all the action profiles are Pareto optima

Whereas this example presents no Nash equilibrium, each of its action profiles is a Pareto optimum. Whatever the present situation is, changing action, for any of the players, would mean damage either her own outcome, or the other player's.

4 Sound web services collaborative mechanism

The problem we handle has been introduced in Chapter 1. In this Chapter, we present the latter in a more specific and detailed way in Section 4.1. A specific modeling is presented in Section 4.1.1.

The introduction of our solution to this issue is presented in Section 4.2. Its different key elements are presented and modelled in Sections 4.3, 4.4 and 4.5.

Section 4.6 is dedicated to the game theory analysis of this first solution.

We take into account the conclusions of the game theory analysis and develop our complete solution in Section 4.8.

Contents

4.1 Problem definition

As we saw in Section 2.1.2, web services providing the same service (namely that have the same functional aspects) can be grouped in communities controlled by a master.

If a web service has not enough requests, or too many, it can be interested in joining such a community. Requests received by communities are dispatched among the different web services composing the community, assuring that none of those web services are either overloaded or idle. A web service wanting to join a community asks the master of the latter for the permission to join. The master can accept or deny the joining request, depending on the needs of the community and on the reputation of the requesting web service.
Community masters can also be interested in inviting new web services in the community. First, if there are too many requests to handle, it is necessary to add new web services to avoid overloading. Second, if there are not enough requests, bringing new web services can bring new clients and new requests. As communities' reputation depends partly on the reputation of the web services composing them, masters can also invite new web services if they find the reputation of those web services interesting for the community.

In order to make a decision, to decide whether or not to accept a requesting web service, the community master has to analyse the web service to know whether or not its reputation is satisfactory. Indeed, if the reputation of a web service does not match the community expectations, bad feedbacks provided because of this web service can have a bad impact on the community reputation.

A solution to get information about web services' reputation is to ask it to an information service, namely a special web service which task is to gather and sell information about other web services' reputation. Masters need to verify the information before making decisions, therefore they send queries about the same web service to several information services and compare their answers.

When providing information to community masters, information services have the choice between telling the truth (and providing the reputation they actually know) and lying (and providing a reputation different from the one they know). Both actions have incentives. On one hand, telling the truth assures the information service to keep a trustful relationship with community masters, but if the information service tells the truth about a bad web service, the latter will not reward the information service for the bad review. On the other hand, lying can provide to the information service bigger direct reward, but contributes to damage the relationship between the information service and the community. As a result, communities have less trust towards lying information services, and ask

less frequently for interactions with them, bringing in fewer rewards.
Furthermore, as masters of communities use several information services at the same time in order to compare their information, the decision to lie or tell the truth also depends on other information services behaviour.

The problem we study is the situation where web services pay information services each time they get chosen to join a community. A web service that does not have a reputation high enough to be accepted in a community could be tempted to cheat and pay more the information services. This bonus reward encourages information services to lie when informing a community, referencing a web service as having a bigger reputation than it actually has. As a result, masters might invite potentially bad web services to join their community, and by doing so, they could damage the reputation of the community.

If the reward web services give to information services to fake their opinion is big enough, these information services are tempted to lie, as there is more direct profit to gain by lying than by telling the truth. If a community accepts a web service because of fake information, the community master expects from it more than it can really handle. After a period of use, community masters can know whether or not the information services have been honest. Indeed, the value of this honesty can be determined by comparing the reputation provided by information services and the actual reputation of the web service. If an information service has not been honest, it is unlikely that the community uses it again as an informant. The opportunity for the information service to be paid for information decreases at the same time.

Information services are in the same situation as *players* in *game theory*. As we saw in Chapter 3, players in game theory have different possibilities regarding the decision or action to make, and are in a game with other players. The outcome of the game represents a gain (positive, negative or neutral) for each player. Each player's action has an impact on other players' outcome, and only the combination of all players' actions defines which outcome will come out the game. Information services, when deciding between telling the truth or lying, have to consider the effects of other information services' action on their own.

If information services have incentives to lie, this can compromise the global integrity of the network (composed by web services, communities of web services, information services and final users). Indeed, communities can neither trust information services nor web services and users can see their requests handled in an unsatisfying way they did not expect.

4.1.1 Modeling

The fact that information services have the possibility to lie is thus a crucial issue. In this Section, in order to facilitate comprehension and reading we present its modeling.

We formalise the problem as follows :

- A **community** of web services i is referred to as C_i
- Two different web services types are taken into account:
  - A **single web service** j, wanting to be referenced and signed up into communities, is referred to as S_j
  - An **information service** k, which get paid by S_j to improve the reputation value of the latter when informing C_i and by C_i to provide information, is referred to as I_k

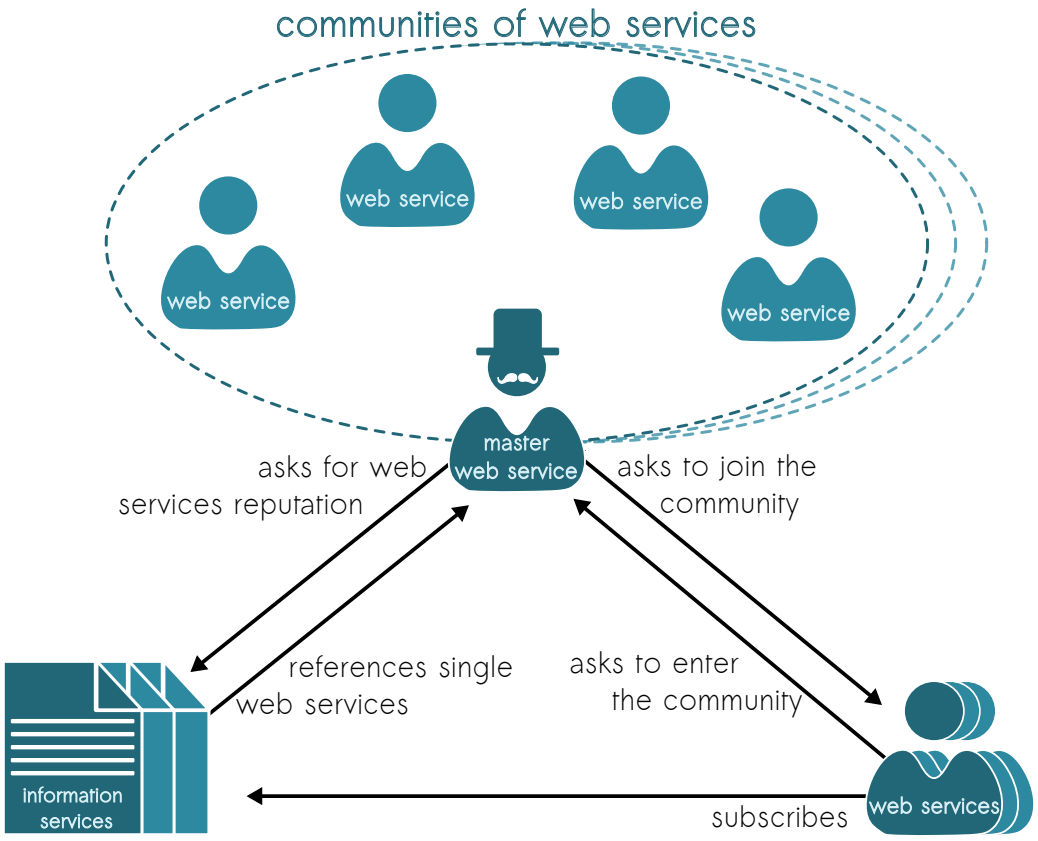

Figure 4.1: Model architecture

Each part of the architecture has different goals :

- Single web service
  - Handle user requests according to its capacity (the single web service wants neither to be overloaded nor to be idle)
  - Be rewarded for a service
  - Have a good reputation
- Community of web services
  - Handle user requests according to its capacity (the community of web services wants its web services neither to be overloaded nor to be idle)
  - Be rewarded for a service
  - Have a good reputation
- Information service
  - Be rewarded to reference single web services

When C_i asks I_k for information about the quality of S_j, I_k provides a reputation value as an answer. A value (representing what I_k reports to be the reputation of S_j to C_i) is assigned to the triplet (C_i,S_j,I_k). We name it $r(C_i, S_j, I_k)$.

Definition 9. *The value $r(C_i, S_j, I_k)$ represents the reputation value of S_j reported by the information service I_k to community C_i.*

Such a value is saved by C_i in registries because C_i needs to keep a record of information received about joining web services in order to compare it to the actual reputation value.

On one hand, the objective of C_i is to have a good reputation and by definition to be composed by reputable web services. C_i would like to encourage I_k to tell the truth, in order to invite only really reputable services and penalise I_k if it is not honest. On the other hand, the objective of S_j is to be invited in a (reputable) community, therefore S_j can give reward to I_k if it gives good (but fake) references.

Profits for I_k can come from different sources, so that the truthfulness of the information is not ensured. The objective of the following section is to look for a solution where the information reflects the truth.

4.2 Solution modeling

In this section, we explain our research for a solution to the problem presented in Section 4.1.

The different concepts we use to build our solution model are presented in the following sections : environment entities in Section 4.3, payments in Section 4.4 and concept of trust and trust evolution in Section 4.5.

Using game theory, we show in Section 4.6 what are the effects of those concepts. A dichotomic approach is used to explore the different possible cases, beginning with simple ones and, browsing in a rational way case by case, evoluting in complexity.

This vision of the environment points out what the defects are and directly shows what should be introduced into the model in order to correct them.

4.3 Environment entities

Those entities have been defined in Section 4.1.1, we add here precisions about the hypothesis we make to build our solution model.

4.3.1 Communities of web services

At first, community masters are honest, their payment reflect the actual experience they had with web services. A community having a good interaction with a web service will not pretend it was bad in order to give less rewards.

Secondly, each community has expectations regarding the web services it adds. We translate those expectations by using a threshold. This threshold represents the minimal reputation value a web service has to offer in order to be accepted by this community. Community masters accept any web service which reputation exceeds the threshold. A community does not have a “maximal threshold”, meaning that no web service is “too good” for a community.

Finally, a community learns from its experience : if the community realises that a information service lied, it will be less inclined to request this information service again. Conversely, if the information service gave to the community an accurate idea of the web service reputation, the community will be motivated to request this information service again.

4.3.2 Web services

Web services can be either *good* or *bad* for a community according to the threshold of the community.

Definition 10. *The value of the judgement of community C_i considering web service S_j is calculated as follows :*

$$ju(S_j, C_i) = \begin{cases} \textbf{\textit{good}} & \text{if } R_{S_j} \geq t_{C_i} \\ \textbf{\textit{bad}} & \text{if } R_{S_j} < t_{C_i} \end{cases}$$

where :

- *t_{C_i} is the value of the threshold for community C_i.*
- *R_{S_j} is the reputation value of S_j.*

4.3.3 Information services

Information services are players of the games, as the outcome of those games depends on their behaviour. Each of them has two possible actions :

- **Tell the truth**
  That means reporting the actual reputation of a web service (the reputation they know).
- **Lie**
  That means reporting a fake reputation about a web service. In order for the lie to be useful, the fake reputation value has to allow the web service to join a community. Therefore,
  - Let t_{C_i} be the value of the threshold for community C_i
  - Let fr_{S_j} be the fake reputation value of S_j

  To join the community, we need to have $fr_{S_j} \geq t_{C_i}$.

4.4 Payments

We divide the payments information services receive in two categories : payments from communities of web services (α, β, γ) and payments from web services (π).

4.4.1 Community payments

The reward given from the community to an information service is divided in 3 parts. This division is done in order to increase information services' honesty. Indeed, 2 of the 3 payments are calculated according to the given information and its accuracy.

The first payment (α) is given to the information service if the latter accepts to provide information to the community, it is the payment for the interaction. The second payment (β) is paid according to the likeness between the information given by the information service and the average reputation value calculated from all information services' information. Finally, the third payment (γ) is determined by comparing the reputation value given by the information service and the actual reputation observed by the community.

α payment

α is the payment C_i gives to I_k as an incentive to give C_i information about S_j. I_k always gets α in its wholeness as it is the payment for the only fact of providing the information.

β payment

β is the payment C_i gives to I_k after collecting reports about S_j from $I_{0..k}$.
The value of β is based on the average reputation $A(C_i, S_j, I_{0..k})$:

Definition 11. *The average reputation of S_j, computed from the information given by $I_{0..k}$ to C_i, is calculated as follows :*

$$A(C_i, S_j, I_{0..k}) = \frac{\sum_{x=0}^{k} (r(C_i, S_j, I_k) Tr_{C_i}^{I_x}(t))}{\sum_{x=0}^{k} Tr_{C_i}^{I_x}}$$

where :

- *$Tr_{C_i}^{I_x}$ represents the value of trust C_i has towards I_x at a time t (defined in Definition 16).*

In this definition, we can see that C_i already uses its experience when computing the average reputation. Indeed, the reputation value reported by each information service will be balanced by the trust that C_i has towards these information services. This way, information given by information services which already proved to be trustworthy will carry more weight than other information services' when computing the average reputation.

Definition 12. *The value of the β payment is calculated as follows :*

$$\beta_k = f_\beta(|A(C_i, S_j, I_{0..k}) - r(C_i, S_j, I_k)|)$$

where :

- $A(C_i, S_j, I_{0..k})$ *represents value of the average reputation.*
- $r(C_i, S_j, I_k)$ *represents the value of the reputation of S_j reported by I_x.*

Definition 13. *The function f_β is calculated as follows :*

$$f_\beta = \begin{cases} \beta_{max} & \text{if } |A(C_i, S_j, I_{0..k}) - r(C_i, S_j, I_k)| \leq \beta_a \\ min\{\beta_{max}, \frac{1}{|A(C_i, S_j, I_{0..k}) - r(C_i, S_j, I_k)|}\} & \text{if } \beta_a < |A(C_i, S_j, I_{0..k}) - r(C_i, S_j, I_k)| < \beta_b \\ 0 & \text{if } \beta_b < |A(C_i, S_j, I_{0..k}) - r(C_i, S_j, I_k)| \end{cases}$$

where :

- β_{max} *represents the maximum value C_i accepts to give as β payment.*
- β_a *is the maximal difference C_i accepts from I_k between the average reputation and the reputation reported by I_k in order to give β_{max} to I_k.*
- β_b *is the maximal difference C_i accepts from I_k between the average reputation and the reputation reported by I_k in order to give a β payment greater than 0 to I_k.*

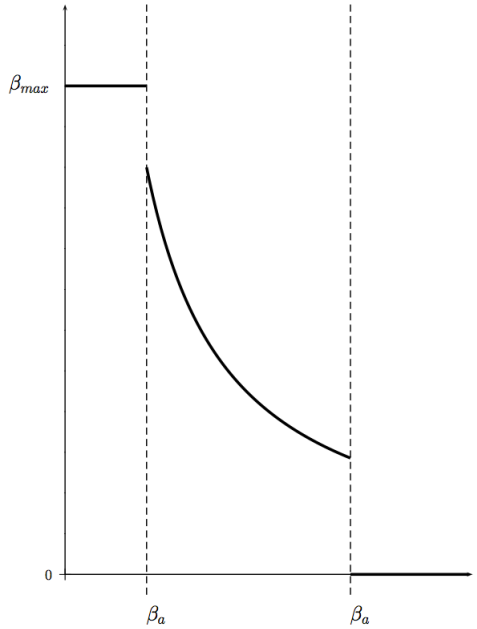

Figure 4.2: β function

γ payment

γ is the payment C_i gives to I_k if S_j joined C_i and C_i assessed its reputation. After this evaluation C_i can compare $r(C_i, S_j, I_k)$ to the observed reputation value $O(C_i, S_j)$ and pays I_k with a value of γ.

Definition 14. *The value of the γ payment is calculated as follows:*

$$\gamma_k = f_\gamma(|r(C_i, S_j, I_k) - O(C_i, S_j)|)$$

where :

- *$r(C_i, S_j, I_k)$ represents the value of the reputation of S_j reported by I_x.*
- *$O(C_i, S_j)$ represents the value of S_j actual reputation observed by C_i when sending requests.*

Definition 15. *The function f_γ is calculated as follows :*

$$f_\gamma = \begin{cases} \gamma_{max} & \text{if } |A(C_i, I_{0..k}, S_j) - r(C_i, S_j, I_k)| \leq \gamma_a \\ min\{\gamma_{max}, \frac{2}{|O(C_i,S_j) - r(C_i,S_j,I_k)|}\} & \text{if } \gamma_a < |O(C_i, S_j) - r(C_i, S_j, I_k)| < \gamma_b \\ 0 & \text{if } \gamma_b < |O(C_i, S_j) - r(C_i, S_j, I_k)| \end{cases}$$

where :

- *γ_{max} represents the maximum value C_i accepts to give as γ payment.*
- *γ_a is the maximal difference C_i accepts from I_k between the observed reputation and the reputation reported by I_k in order to give γ_{max} to I_k.*
- *γ_b is the maximal difference C_i accepts from I_k between the observed reputation and the reputation reported by I_k in order to give a γ payment greater than 0 to I_k.*

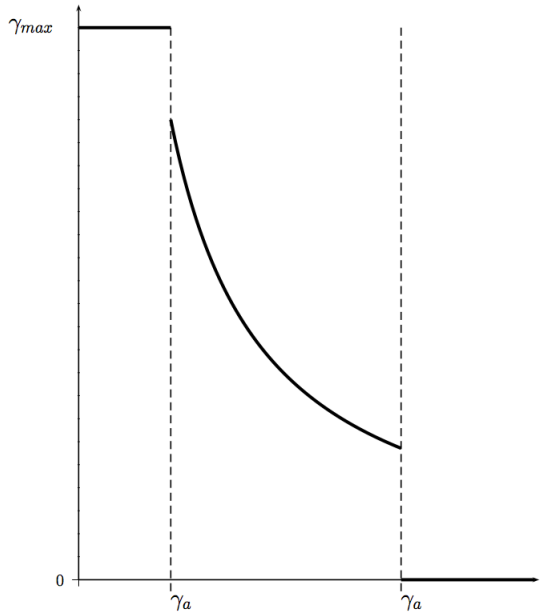

Figure 4.3: γ function

4.4.2 Web services payment

π is the payment S_j gives to I_k to increase the reputation of S_j when reporting about it to communities. This payment will only be received if C_i chooses to add S_j.

4.5 Trust evolution

In order to receive the best information, community masters prefer asking for information to trustworthy information services, that is to say information services that already gave accurate information about web services' reputation. In order for community masters to know which information service it has to choose,

masters keep records of their previous experiences with each information service by means of a trust value related to each information service.

This trust value at a time t, is represented as $Tr_{C_i}^{I_x}(t)$.

Definition 16. *The trust value of community C_i towards I_x at a time t is calculated as follows :*

$$Tr_{C_i}^{I_x}(t) = Tr_{C_i}^{I_x}(t-1) + f_\tau(|r(C_i, S_j, I_k) - O(C_i, S_j)|)$$

where :

- $r(C_i, S_j, I_k)$ *represents the value of the reputation of S_j reported by I_x.*
- $O(C_i, S_j)$ *represents the value of S_j actual reputation observed by C_i.*
- $Tr_{C_i}^{I_x}(t-1)$ *represents the trust at a time $t-1$, that is too say the last known value of the trust.*
- f_τ *is the trust update function.*

Definition 17. *The trust update function is calculated as follows :*

$$f_\tau = \begin{cases} max\ bonus & if\ |O(C_i, S_j) - r(C_i, S_j, I_k)| \leq \tau_a \\ m * |O(C_i, S_j) - r(C_i, S_j, I_k)| + b & if\ \tau_a < |O(C_i, S_j) - r(C_i, S_j, I_k)| < \tau_b \\ max\ malus & if\ \tau_b < |O(C_i, S_j) - r(C_i, S_j, I_k)| \end{cases}$$

where :

- $m = \frac{(max\ malus - max\ bonus)}{\tau_b - \tau_a}$.
- $b = (max\ malus - m * max\ bonus))$.
- *max bonus represents the maximal trust gain C_i gives to I_k.*
- *max malus represents the maximal trust loss C_i imposes on I_k.*
- τ_a *is the maximal difference C_i accepts from I_k between the observed reputation and the reputation reported by I_k in order to add max bonus to the trust given to I_k.*
- τ_b *is the maximal difference C_i accepts from I_k between the observed reputation and the reputation reported by I_k in order to add to the trust given to I_k a value greater than max malus.*

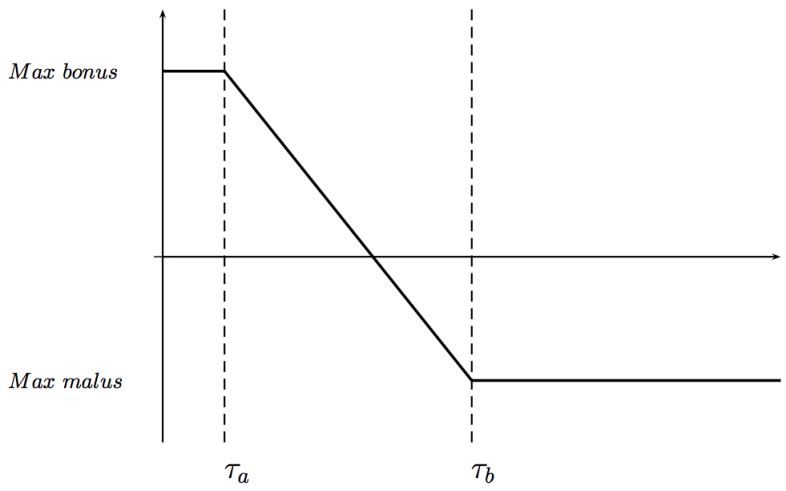

Figure 4.4: τ function

4.6 Game theory study

According to our model, when requested for information about a web service, information services only use payments α, β, γ and π as data to make their decision. Using game theory, we present what are the effects of these payments.

4.6.1 Types of games

We take two types of games into account. On one hand, we consider *one shot game*. As the name implies, the game is there only played once. On the other hand, we consider *repeated games*, where the game is played several times. As a consequence, each game can be influenced by the outcome of the previous ones.

4.7 Cases overview

In this section, we review the different cases we decided to consider. For each of them, we analyse the situation as a game involving two players that are represented by a 2x2 array. Rows show the strategies of a single information service. Columns indicate the strategies of the group of all the other involved information services. Each cell of the array represents an action profile, that is to say the outcome each player has according to their chosen strategies. The first outcome is the one of the single information service and the second is the one of the group of information services. For each case, we try to know whether there is a Nash and/or a Pareto equilibrium.

We begin in Section 4.7.1 by considering the several cases where the single web service is honest. In these cases, the web service does not try to corrupt information services by giving them rewards. Therefore, we only take into account the first three payments α, β and γ. We first see what happens in the context of a one shot game. Then, we move to a repeated game context.

Secondly, in Section 4.7.2 we admit that web services can have dishonest behaviour. In other words, information services can receive π incentives in order to improve fraudulently the information they give to communities. As before, we start by studying a one shot game and then a repeated one.

4.7.1 Honest single web services

- **One shot game**
  - *S is a good web service*

    First, let us assume that the web service S, wishing to enter the community, is a good one.

		Other information services	
		Tell the truth	Lie
Information service	Tell the truth	$(\alpha+\beta+\gamma)$, $(\alpha+\beta+\gamma)$	(α), $(\alpha+\beta_-)$
	Lie	(α), $(\alpha+\beta_-+\gamma)$	$(\alpha+\beta)$, $(\alpha+\beta)$

Table 4.1: Honest single web services - One shot game - S is good

If every information service tells the truth and informs the community that the web service is good, then every information service receives a maximum payment of $\alpha+\beta+\gamma$. Indeed, information services receive α in reward for the processing of the request. They gain β because the value of the report that each information service gives is close to the average of all reported values. Finally, they receive a third payment γ thanks to the fact that the observed reputation and the announced reputation are close.

If one information service decides to change its strategy while all the others remain constant, namely if it decides to lie while the others continue to tell the truth, this information service degrades its total payment. It still receives α but neither β nor γ because the reputation it reported is far from the average and the observed reputation. On the other hand, the other information services also experience a

degradation in their total payment. Indeed, because one information service decided to report a reputation totally different, the average reputation is smaller than if every information service had reported a similar value. We express the fact that the average reputation is smaller by writing β_- instead of β. For those information services, the reported reputation and the observed reputation are alike, therefore the γ payment does not degrade.

If it is the group of information services that decides to change its strategy and lie, information services composing that group only get α and β_- as payment. If the majority of information services announce that the web service is bad, the community does not accept it. Therefore the third payment γ is not granted. The information service that did not change its strategy only gets α. This information service receives neither β_- , because its announced reputation is far from the average one, nor γ, because the web service does not enter the community.

If everyone decides to lie, all information services get α and β and nobody receives γ as the web service does not enter the community.

We can see that there is an incentive to tell the truth for everyone. Telling the truth corresponds to the situation that guarantees the maximum payment $\alpha+\beta+\gamma$. That situation is the only Nash equilibrium of this case and is Pareto optimal.

– *S is a bad web service*

In this second case, we assume that the web service that wants to be part of the community is a bad one.

		Other information services	
		Tell the truth	Lie
Information service	Tell the truth	$(\alpha+\beta)$, $(\alpha+\beta)$	$(\alpha+\gamma)$, $(\alpha+\beta_-)$
	Lie	(α), $(\alpha+\beta_-)$	$(\alpha+\beta)$, $(\alpha+\beta)$

Table 4.2: Honest single web services - One shot game - S is bad

If every player tends to tell the truth and reveals that the web service is bad, they all get a payment of $\alpha+\beta$. They receive α for the processing of the request and β because each information service reports a similar value and is therefore close to the average reputation value.

Nevertheless, the web service does not enter the community and nobody gets the γ payment.

If the information service decides to modifies its strategy and lies by announcing that the web service is good, it degrades its total payment. The information service actually still receives α but not β as the reputation it announced is no more comparable to the average. As for the other information services, they get the two first payments but the second one is slightly lower in comparison to the previous situation. Therefore, they receive $\alpha + \beta_{-}$.

In the next situation, let us assume that the group of other information services changes its strategy and begins to lie. Because the majority of information services declare the web service as good, the web service is accepted to join the community. Soon, the community realises that it was not the truth and that the web service was actually a bad one. Therefore, the group of information services receives $\alpha + \beta_{-}$. On the other hand, the information service that kept its strategy of telling the truth gets $\alpha + \gamma$. Indeed, the information service does not receive the second payment but is rewarded by γ as it reported correctly that the web service was bad.

If everyone decides to lie, all the information services get $\alpha + \beta$. Nobody receives γ because the web service joins the community but the latter discovers that the web service was actually bad.

In this case, we can see that the strategy of telling the truth for every information services is the only Nash equilibrium. But there are two Pareto optimal situations : one if everybody tells the truth and another if each information service lies.

- **Repeated game**
  - *S is a good web service*

In the context of a repeated game, if the web service that wishes to enter the community is good, the solution is trivial. Indeed, if the best strategy for the information services is already to tell the truth in an one shot game, no matter how many times the game is played, the players will not change their behaviour.

– *S is a bad web service*

If the web service is bad, the argumentation is the same than in the previous case. Because the information services already tend to tell the truth in an one shot game, they will not change their strategies no matter how many times the game is played.

4.7.2 Dishonest single web services

We now assume that web services can have a dishonest behaviour. In other words, we mean that a web service can give a π incentive to information services in order to make them fake their reputation and provide an improved reputation value to the communities.

- **One shot game**

  – *S is a good web service*

  If the web service that wants to join the community is a good one, we assume that it has no interest in trying to corrupt information services as it already has a good reputation. Therefore, this case is similar to the one with a honest web service. We have a Pareto optimal Nash equilibrium when all the Information services tell the truth.

  – *S is a bad web service*

  Let us assume now that the web service that wishes to enter the community is bad and that it can give a π incentive to information services in order to make them provide a fake improved reputation value to the community.

		Other information services	
		Tell the truth	Lie
Information service	Tell the truth	$(\alpha+\beta)$, $(\alpha+\beta)$	$(\alpha+\gamma)$, $(\alpha+\beta_-+\pi)$
	Lie	(α), $(\alpha+\beta_-)$	$(\alpha+\beta+\pi)$, $(\alpha+\beta+\pi)$

Table 4.3: Dishonest single web services - One shot game - S is bad

We analyse the situation in which every information service tells the truth and reports to the community that the web service is bad. Here, each information service receives $\alpha+\beta$ as total payment. α is once again given in exchange of the processing of the request. Information

services also get β because they all report a similar reputation value for the web service and are therefore all close to the average reputation value. They do not receive γ because the web service does not join the community, as it has been reported as a bad one.

If the information service decides to change its strategy and begins to lie, its total payment decreases to α. Indeed, the information service does not receive β anymore because it is the only one to report a good reputation value and is therefore far from the average. As for the other information services, they receive $\alpha + \beta_{-}$. Nobody gets γ because the web service does not join the community.

In the situation where the group of other information services changes its strategy and lies, the web service joins the community. The single information service that continues to tell the truth receives α but not β. Nevertheless, the information service gets γ for having truthfully reported that the web service was not a good one. The other information services receive $\alpha + \beta_{-}$. But because the web service joins the community, they also get the π reward from the web service itself.

If all information services decide to lie, they get $\alpha + \beta + \pi$. The web service joins the community but the community master will discover that the web service is not as good as expected. Therefore, information services do not receive γ. On the other hand, because of their lie, they are rewarded by π.

In this case, we see that lying and obtaining the π reward from the web service for all the information services corresponds to the unique Nash equilibrium of this game. This situation is also Pareto optimal.

- **Repeated game**

  - *S is a good web service*

    Once again, if the web service that wants to enter the community is good and the game is played repeatedly, the best strategy for information services is to tell the truth because it was already the case in the one shot game and nothing changes this behaviour.

  - *S is a bad web service*

The problem we face in the one shot game with a bad web service is that the best strategy for the information services is to take the incentive π and lie to the community. Obviously, this should not be an acceptable behaviour in the long term and we therefore need to find a solution in order to prevent this from happening.

4.8 Introduction of the trust value into the choice process

Giving a clear motivation to lie puts the information service in a situation where it has to choose between different actions, each creating different outcomes for the information service. The data the choices are based on are various. An information service has to consider the direct gain it can earn from its choice. At least a small part of this gain comes from the community that requested it, but a part of the gain can also be composed of bribes coming from the web service.

If we assume that only those payments were taken into account, the choice would be simple : the action to choose is the action that brings the bigger gain to the information service.

But, as we saw in Section 4.3.1, communities learn from their experience and from their mistakes.
If an information service lies to a community, motivated by a consequent bribe, admittedly it will get a large gain instantaneously, but will loose credibility with regard to the community. The latter will be no more, or at least less motivated to ask this particular information service for information about web services. Fewer requests means less opportunities to get paid for information services. As a consequence, an information service will need to consider the direct gain but also the effects of its action on the motivation for communities to request it.

To combine payments seen in Section 4.4 and this idea of keeping being paid in the future, we propose in this work to introduce a solution by means of a *reward function* information services will use when having to decide whether or not they should lie. The output of the function will be a value representing direct payments, but also expectation about future payments. This reward function, in order to reflect this expectation about future payments, has to balance direct payments with a parameter related to the possibility of getting those future payments.

The trust a community has towards an information service is the value the community will consider when requesting this information service, and thus pay it for the service. This is why, when creating our reward function, we decide to use the trust as the balance factor.

Definition 18. *The reward function for an information service I_k, requested among other information services $I_{0..z}$ by community C_i to give information about S_j can be calculated as follows :*

$$\begin{aligned} R(C_i, S_j, I_k, I_{0..z}) = & \quad \alpha(k) + \beta(k) + \gamma(k) + \pi(k) \\ + & \quad Tr_{C_i}^{I_k}(t+1) * (\alpha(k) + \beta(k) + \gamma(k) + \pi(k)) \end{aligned}$$

where :

- $Tr_{C_i}^{I_k}(t+1)$ *is the value of the trust ot C_i towards I_k after the choice of I_k.*

The first part of the function corresponds to the direct value a player gains from the four possible payments α, β, γ and π.

The main idea of the second part of the function is that $Tr_{C_i}^{I_k}$ will vary depending on the behaviour of the information service. If the information service provides fake reputation reports, it is possible that a community will allow a bad web service to join. The community realises it was a lie and the information service gets penalised by a decrease of $Tr_{C_i}^{I_k}$. On the other hand, if the information service acts honestly and allow a good web service to enter a community, then $Tr_{C_i}^{I_k}$ increases.

Thanks to that mechanism, the second part of the payoff function acts as a bonus/malus depending on the behaviour of the information service. The purpose for the information service is obviously to maximise the possible payoff bonus. That can only be achieved by adopting a honest behaviour, so in other words, by telling the truth on a repeated basis.

The reward function acts as a utility function for the information service. Indeed, the value of $Tr_{C_i}^{I_k}(t+1)$ depends on the information the information service gives (a true information or a lie). Therefore the value of the reward the information service gets will also depend on it. When having to decide whether it is best for it to lie or tell the truth, an information service can compare the different possible values of the reward function C.

In order to make the value of the reward function closer to the definition of a utility function value, we decide to bound it between 0 and 1.
The actual value of the reward function is $R(C_i, S_j, I_k, I_{0..z}) \in [0, 2 * (\alpha(k) + \beta(k) + \gamma(k) + \pi(k))]$.
By introducing the division of this value by $2 * (\alpha(k) + \beta(k) + \gamma(k) + \pi(k))$, we limit the range to $[0, 1]$.

Definition 19. *The reward function for an information service I_k, requested among other information services $I_{0..z}$ by community C_i to give information about S_j can be calculated as follows :*

$$R(C_i, S_j, I_k, I_{0..z}) = \frac{\alpha(k) + \beta(k) + \gamma(k) + \pi(k) + Tr_{C_i}^{I_k}(t+1) * (\alpha(k) + \beta(k) + \gamma(k) + \pi(k))}{2 * (\alpha(k) + \beta(k) + \gamma(k) + \pi(k))}$$

where :

- $Tr_{C_i}^{I_k}(t+1)$ *is the value of the trust ot* C_i *towards* I_k *after the choice of* I_k.

5 Simulation tools

Now that we propose a solution for the problem we approached, we demonstrate that the latter is correct. In order to do that, we use a program that allows us to simulate the different entities we used and their behaviour in the way we defined them. The results of this simulation program should assess whether the theoretical solution we presented in Section 4.2 is coherent with what we can expect in a real-time context.

Contents

5.1 Context and existing tools

In order to conduct experimentations, we first consider some of the existing simulation tools, *Ascape* [Asc11] and *MASON* [MAS11] in particular. Nevertheless, some limitations force us not to use these two simulators and implement our own solution instead.

As explained previously, one of the main issues in web services and communities of web services networks is for the community master to choose whether or not to accept a web service in the community. As an answer to this problem, we introduced the new concept of information service which role is to inform communities about web services' reputation.

The concept of information service being the main component of our system, we needed it to be part of the simulation. Unfortunately, because of the originality of this concept, it was logically impossible to find it in an existing simulator. Another crucial element of our model was the introduction of a specific way of calculation for the payments and the trust evolution.
Because our model introduces new and targeted concepts, we decided that a new specific tool would be more appropriate to reach the objectives and details of our model.
As explained in A.1, even our specific simulator requires some implementation choices and slight restrictions. Reaching an appropriate solution with an existing simulation tool would thus have represented a huge amount of additional modifications.

5.2 Tool presentation

In this section, we make a brief presentation of the simulation tool we created. A more complete and technical presentation of the tool can be found in the Appendix A.

5.2.1 Environment and entities

In the simulation, three types of entities interact with each other in order to reach their goals. *Web services* want to join communities in order to get more requests. *Communities of web services* want to add to the community only web services that have a reputation that fits their expectations. *Information services* want to get the bigger reward possible when giving information to communities.

Here follows a quick presentation of our modeling of those entities into the simulation.

Each community of web services knows some information services and requests a certain number of them when looking for information about web services. Communities trust information services they know. The trust value attributed to each information service depends on the validity of previous interactions between the community and these information services. Logically, when looking for information, communities request the information services they trust the most.
As presented in Definition 10, the judgement of a community towards a web services depends on the community threshold. Each community has a specific threshold contained between 0 and 1.

Web services are presented in the simulation as entities defined by a quality of service and a quality of service variability. The quality of service of a web service is the value of the maximum quality of service a web service is able to offer. When working, a web service cannot always offer this maximal quality of service. If it is overloaded, for instance, the speed at which the web service provides an answer to a request decreases, affecting the quality of service. In our simulation, we do not simulate the actual sending of requests. The quality of service variability is thus a parameter we use to define the probability for a web service to answer to a request with a quality of service different from its maximum capacity.

Because we do not simulate users, the representation of the reputation in the simulation is quite simple. The reputation of a web service is based on a general opinion of its quality of service. Therefore, an information service knowledge about a web service reputation is computed as the average value of a set of quality of service values.

The more honest an information service is, the more likely it is that communities request it to get information about web services. On the contrary, if an information service is used to lie, communities do not ask it for information. Consequently, if an information service has not been requested for a long time, it is preferable for it to tell the truth if asked to handle future requests. Therefore, if an information service is not requested for a defined amount of time, its probability to tell the truth increases. This probability is expressed with a parameter we name *truth trend*.

5.2.2 Rewards and incentives

Rewards and trust are granted if the information provided by an information service is close to a specific value (the average reputation value for the β function and the actual reputation value for both γ function and the trust update function). The closer the information is to this specific value, the bigger is the reward. In other words, the smaller is the difference between the information and this specific value, the bigger is the reward.

In order to compute the different rewards and the trust update, we use a mechanism of *two points function*. The β function, the γ function and the trust update function are divided in three sections.

If the difference between the given information and the referenced value is in the first section (between 0 and the A point), the reward (or trust update) is maximum. If the difference is in the second section (between points A and B), the reward (or trust update) is computed according to the specific function (see Definitions 13, 15 and 17). Finally, if the difference is in the third section (greater than point B), the reward (or trust update) is minimal.

5.2.3 User interface

We developed our simulation tool so that the behaviour of each of these entities can be configured. In order to make this process easier, we created a user interface. The different parameters that can be set are grouped in two configuration screens.

In Figure 5.1, the user can modify the parameters related to the different entities. The number of communities can be chosen as well as, for each of them, the minimum reputation level that a web service must have in order to be accepted in the community.

The user can select the number of web services and the number of categories in which web services will be distributed. For each category, a percentage of

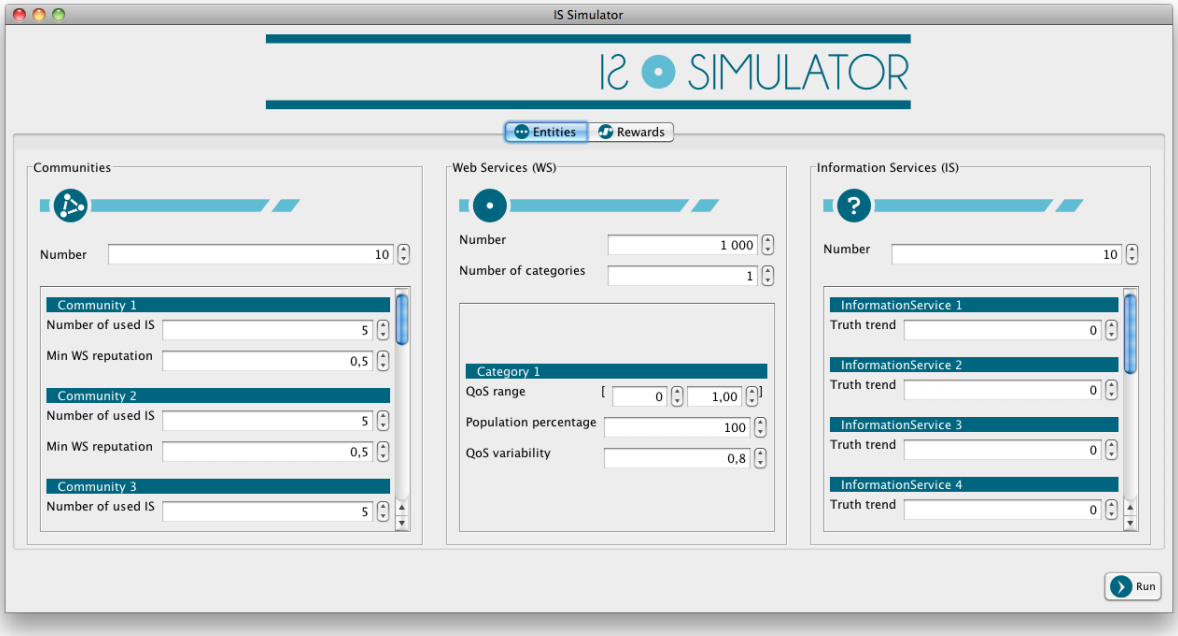

Figure 5.1: User interface - Entities configuration

population and a quality of service range can be chosen.

The number of information services can be selected and, for each information service, an initial truth trend value.

In the second configuration screen of the simulation tool, shown in Figure 5.2, the parameters related to the rewards and incentive can be defined by the user. The behaviour of the different functions can be modelled as the user can choose the extreme points (below point A, the gained value will be maximum whereas above point B, the gained value will be minimum).

5.2.4 Results

When the user starts the simulation, the evolution of the results is displayed using different graphs. The latter can be saved on the computer. A progress bar indicates the progression of the current simulation. Information about events happening during the simulation are also shown in a notification area.

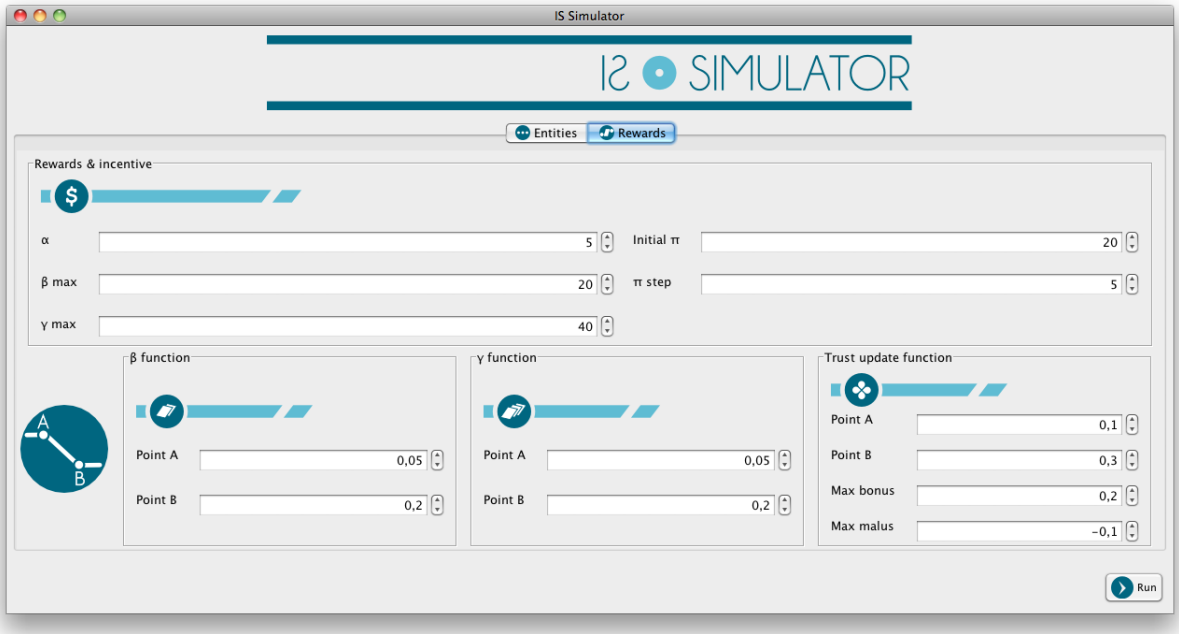

Figure 5.2: User interface - Rewards and incentive configuration

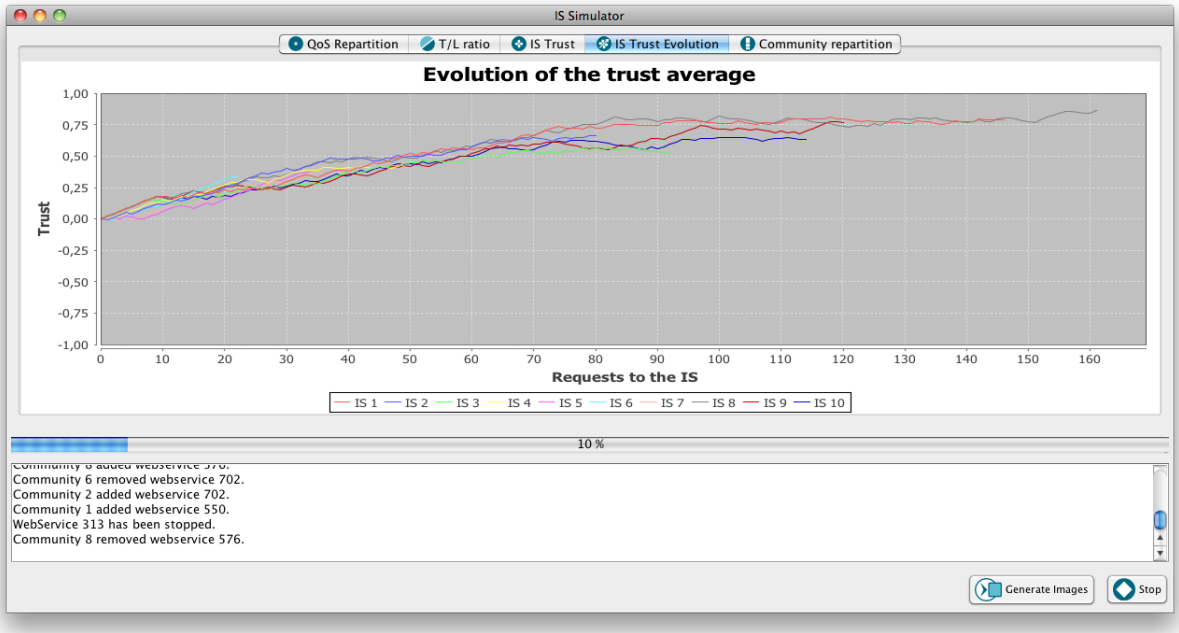

Figure 5.3: User interface - Results

6 Simulation results

In Section 4.8, we proposed a theoretical solution to insure that information services in the network would have an honest behaviour. Thanks to an appropriate adjustment of incentives and rewards and the consequent trust management, we model a mechanism where information services are motivated to tell the truth when communities request information.

In this chapter, we use the simulation tool we developed (presented in Chapter 5 and detailed in Appendix A) and its results to show the validity of the theoretical concepts we introduced. The main objective of this experimentation is to illustrate the theoretical development by simulating the evolution of several test environments.

In the first section of this chapter, we introduce the simulation test environments. We present each of these environments and detail the various parameters we set for each of them. Results of each test environment are then examined and compared according to various outputs. We mainly focus on three aspects. First, we analyse the repartition of web services in the communities at the end of each run, that is to say the percentage of good web services that were accepted, of bad web services that were accepted and of bad web services that were rejected. We then take a look at each information service trust average evolution. Finally, the Truth/Lie ratio evolution is studied.

At the end of this chapter, we summarise the results we obtained by doing a comparative analysis of the data we gathered from the simulation of each test environment.

Contents

6.1 Test environments

In this section, we review the different test environments we use during the simulation. Among all possible environments, we selected the seven most interesting ones. First and second environments are used as basis. We then present the variations we introduced in order to obtain the other five environments.

- Environment 1: Basic environment without R

  To obtain this first environment, we present the situation in which the reward function (Definition 19) is not used within the games. The trust function (Definition 16) is still used by communities to update the trust they have towards information services, but information services do not take the impact on their trust into account when choosing whether to lie or tell the truth.

  In this environment, the outcome of a game for a information service is calculated as follows :

  $$\alpha(k) + \beta(k) + \gamma(k) + \pi(k)$$

  instead of

  $$\frac{\alpha(k) + \beta(k) + \gamma(k) + \pi(k) + Tr_{C_i}^{I_k}(t+1) * (\alpha(k) + \beta(k) + \gamma(k) + \pi(k))}{2 * (\alpha(k) + \beta(k) + \gamma(k) + \pi(k))}$$

  This aims to compare benefits we can obtain in terms of results satisfaction when we use and when we do not use R.

  We take a sample of 1000 web services. As the time length of a simulation depends on the number of used web services, this number is big enough to give us results consequent enough to observe the evolution of the trend. A bigger number of web services would not have improved the quality of the results but only impacted the duration of tests.

  The population of these web services is composed of only one category of web services whose QoS ranges from 0 to 1 according to a random repartition. We assume that the web services are relatively small and can therefore not handle a lot of requests. As a result, it is likely that a request is not processed with the maximal QoS, therefore the QoS variability of this unique

category has been set to 0.8.

Regarding information services, this first environment counts 10 of them. This number insures the communities have a sufficient choice when they request information about web services without being too random and scattered. Their initial truth trend has been set to a neutral value of 0.0. This value expresses the fact that at the beginning of the simulation, information services do not have personal incentive to tell the truth.

This first environment is composed by 10 communities. Each community uses 5 information services when evaluating the quality of a particular web service. To let a web service join, all communities require a minimum reputation value of 0.5.

Concerning payments and rewards, we first set the value of α to 5.0, β max to 20.0 and γ max to 40.0. The β and γ functions used in this environment are shown in Figures 6.1 and 6.2. The initial π is set to 20.0 and the π step to 5.0.

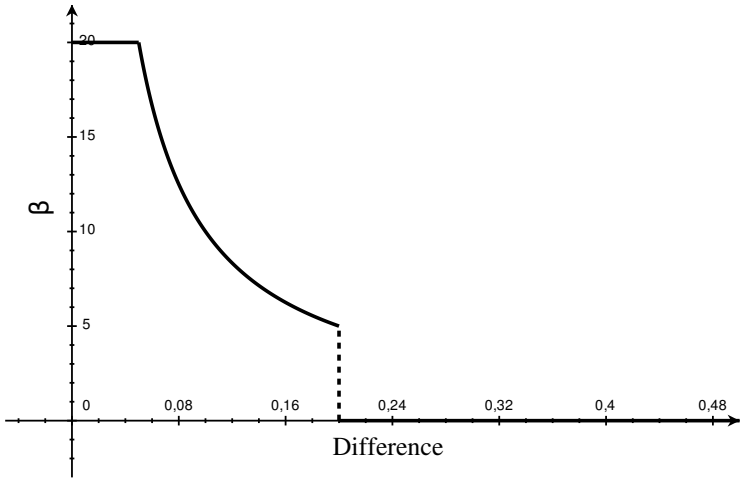

Figure 6.1: Environment 1 - β function

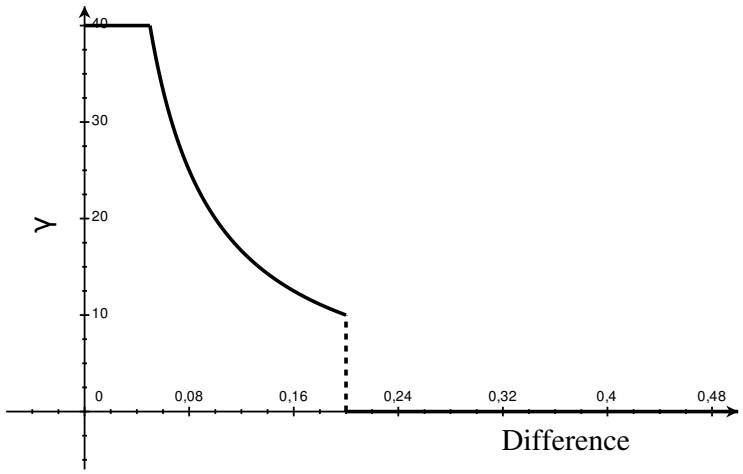

Figure 6.2: Environment 1 - γ function

Finally, the trust update function is set as shown in Figure 6.3 with a max bonus value of 0.2 and a max malus of -0.1.

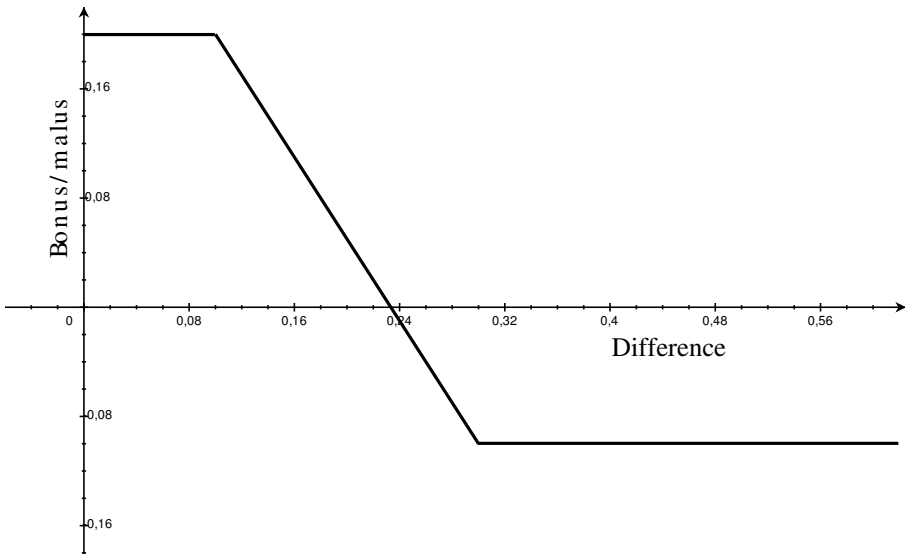

Figure 6.3: Environment 1 - Trust update function

- Environment 2 : Basic environment with R

  In this second environment we use the same parameters than those set for the first one. The only difference is that, from now and for every other test environment, the reward function is used.

  We take a sample of 1000 web services as it provides results that allow good observations without having a too long simulation.

  The population of these web services is composed of only one category of services whose QoS ranges from 0 to 1 according to a random repartition and with a QoS variability of 0.8.

  The second environment features 10 communities that use 5 information services when they request information about a web service. Information services start with a neutral initial truth trend of 0.0.

  Regarding payments and rewards, we set first the value of α at 5.0, β max at 20.0 and γ max at 40.0. The β and γ functions used in this environment are shown in Figures 6.4 and 6.5. The initial π is set to 20.0 and the π step to 5.0.

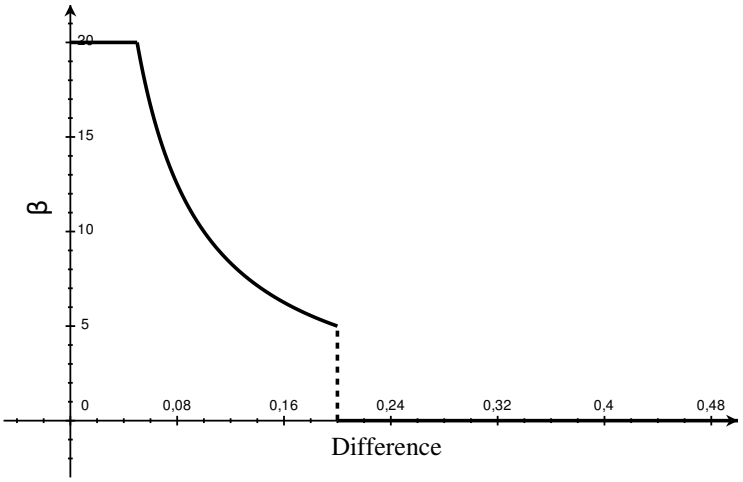

Figure 6.4: Environment 2 - β function

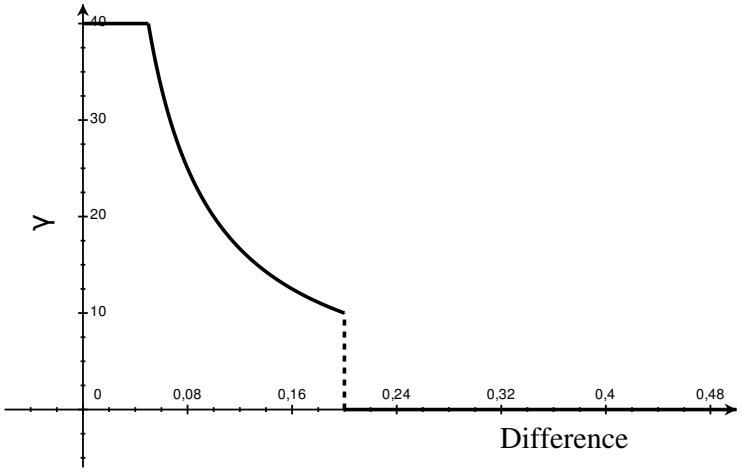

Figure 6.5: Environment 2 - γ function

The trust update function used with this environment is set as shown in Figure 6.6 with a max bonus value of 0.2 and a max malus of -0.1.

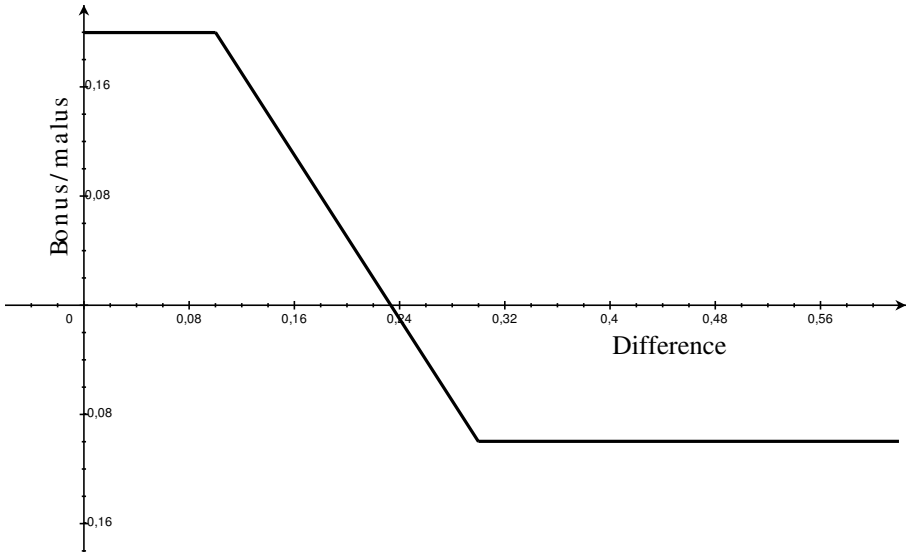

Figure 6.6: Environment 2 - Trust update function

- Environment 3 : Increased initial π and π step

  This third environment is very similar to the second one except for the value of the π payment.

  We take a sample of 1000 web services as it provides results that allow good observations without having a too long simulation.

  The population of these web services is composed of only one category of services whose QoS ranges from 0 to 1 according to a random repartition and with a QoS variability of 0.8.

  The third environment features 10 communities that use 5 information services when they request information about a web service. Information services start with a neutral initial truth trend of 0.0.

  Regarding payments and rewards, we set first the value of α at 5.0, β max at 20.0 and γ max at 40.0. The β and γ functions used in this environment are shown in Figures 6.7 and 6.8.

  The main change of this environment compared to the second one is that we increase both π related parameters so that the initial π is set to 40.0 and the π step to 10.0. Thanks to these modification, we should see that information services are more tempted to lie.

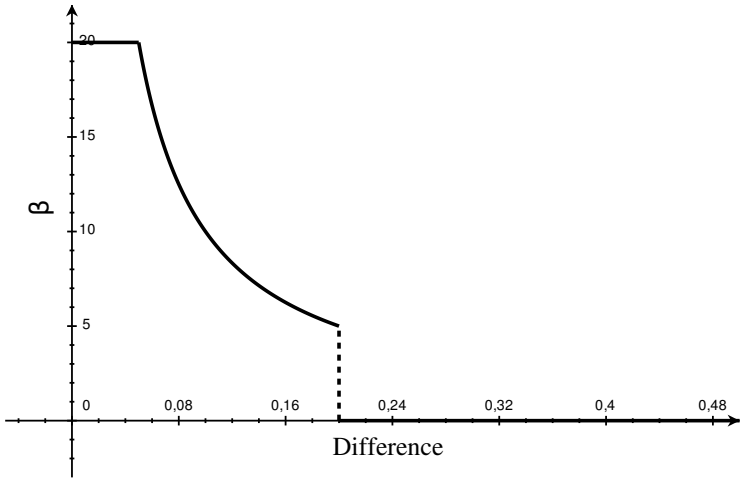

Figure 6.7: Environment 3 - β function

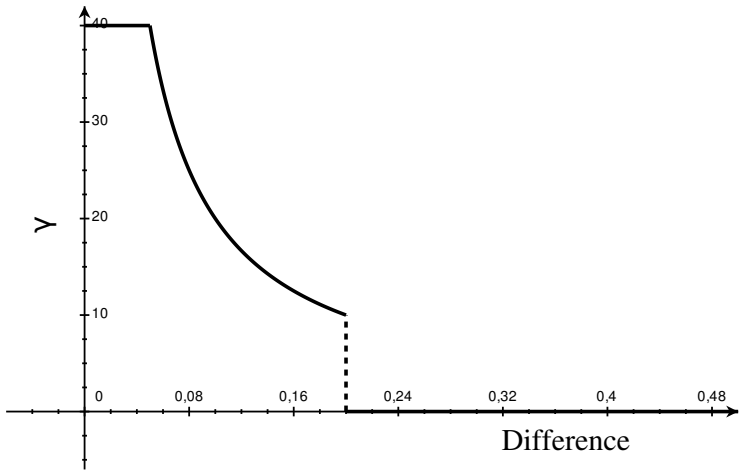

Figure 6.8: Environment 3 - γ function

The trust update function used with the environment is set as shown in 6.9 with a max bonus value of 0.2 and a max malus of -0.1.

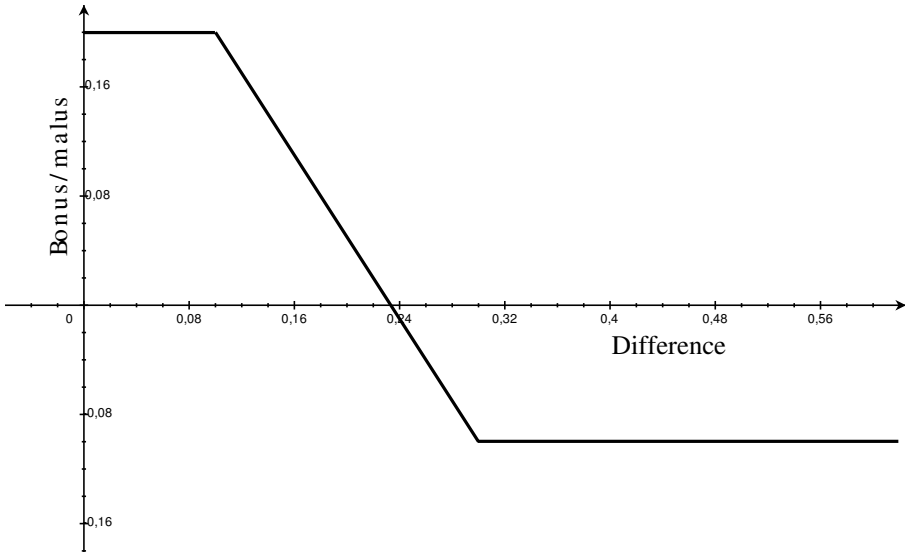

Figure 6.9: Environment 3 - Trust update function

- Environment 4 : Increased maximum malus

  In this fourth environment, we took the parameters of the third environment as basis but introduced some modifications.

  We take a sample of 1000 web services as it provides results that allow good observations without having a too long simulation.

  The population of these web services is composed of only one category of services whose QoS ranges from 0 to 1 according to a random repartition and with a QoS variability of 0.8.

  The fourth environment features 10 communities that use 5 information services when they request information about a web service. Information services start with a neutral initial truth trend of 0.0.

  Regarding payments and rewards, we set first the value of α at 5.0, β max at 20.0 and γ max at 40.0. The β and γ functions used in this environment are shown in Figures 6.10 and 6.11. The initial π is set to 40.0 and the π step to 10.0.

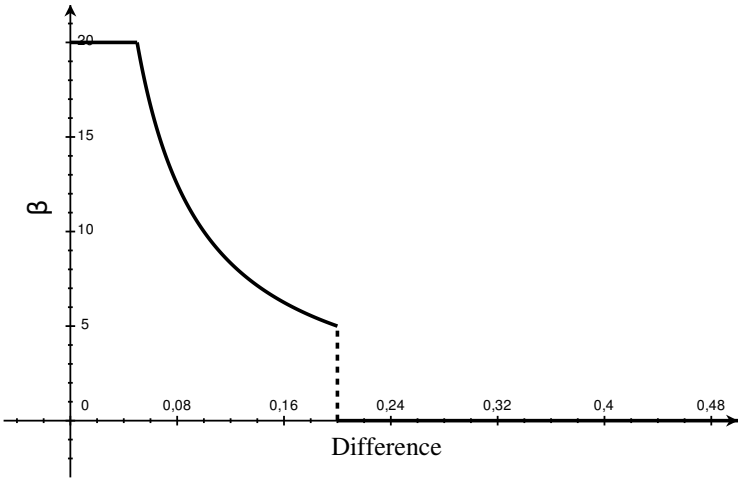

Figure 6.10: Environment 4 - β function

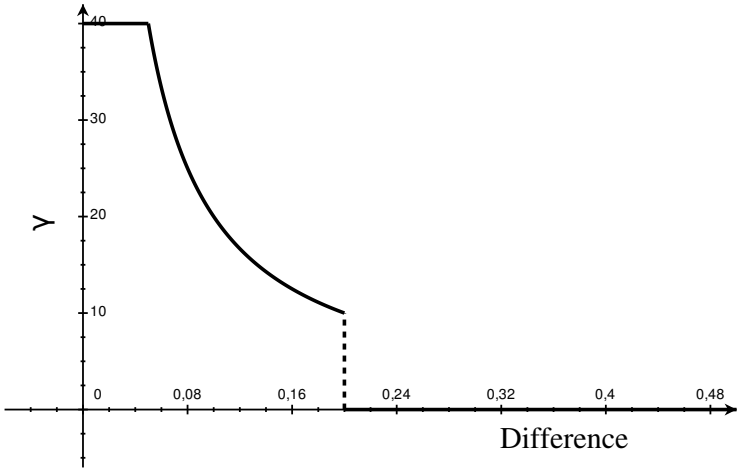

Figure 6.11: Environment 4 - γ function

In this fourth environment, we modified the shape of the trust update function. We set the value of the maximum malus at -0.5. The trust update function here is represented by the Figure 6.12. This modification should incite the information services to be more honest.

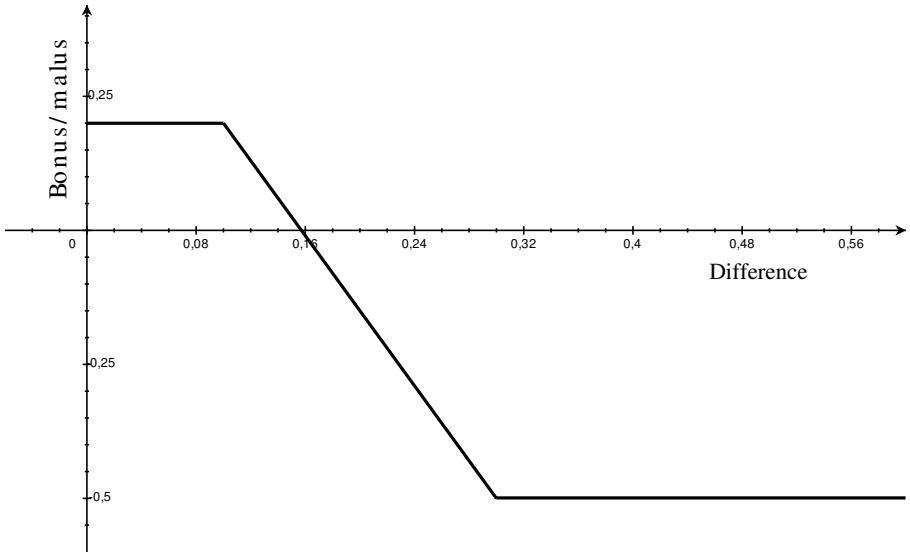

Figure 6.12: Environment 4 - Trust update function

- Environment 5 : Increased maximum bonus

  In this environment, we also took the third environment as basis but we modified it in a different way than in the fourth environment.

  We take a sample of 1000 web services as it provides results that allow good observations without having a too long simulation.

  The population of these web services is composed of only one category of services whose QoS ranges from 0 to 1 according to a random repartition and with a QoS variability of 0.8.

  The fifth environment features 10 communities that use 5 information services when they request information about a web service. Information services start with a neutral initial truth trend of 0.0.

  Regarding payments and rewards, we set first the value of α at 5.0, β max at 20.0 and γ max at 40.0. The β and γ functions used in this environment are shown in Figures 6.13 and 6.14. The initial π is set to 40.0 and the π step to 10.0.

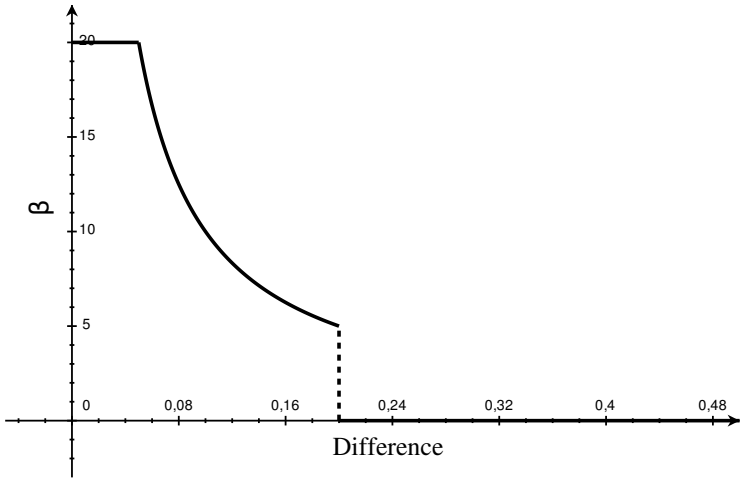

Figure 6.13: Environment 5 - β function

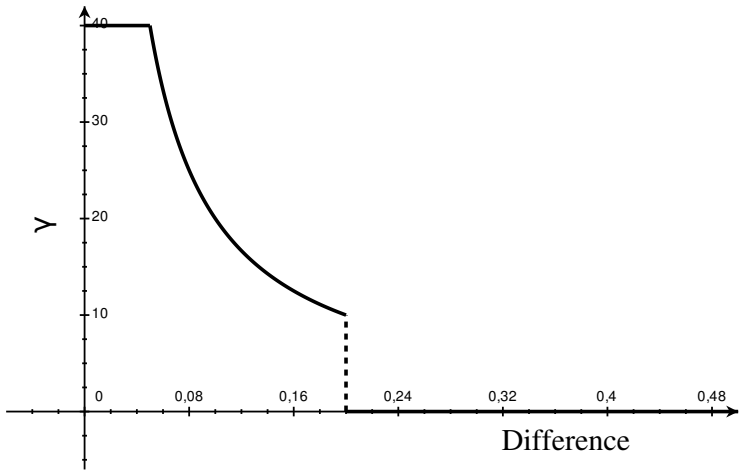

Figure 6.14: Environment 5 - γ function

The main difference in comparison to previous environments lays in the variables of the trust update function. Indeed, we decide this time to use a max bonus of 0.5 instead of an important max malus. The used trust update function is shown in Figure 6.15. With this modification, the information services should be more tempted to tell the truth.

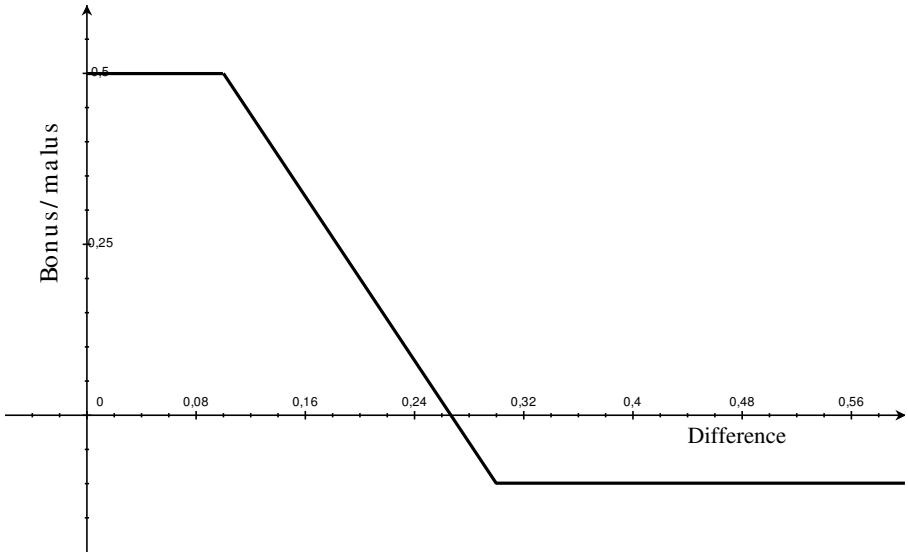

Figure 6.15: Environment 5 - Trust update function

- Environment 6 : Low number of used information services

  In this sixth environment, we come back to the second environment as basis but modify the way communities handle their relation with the information services.

  We take a sample of 1000 web services as it provides results that allow good observations without having a too long simulation.

  The population of these web services is composed of only one category of services whose QoS ranges from 0 to 1 according to a random repartition and with a QoS variability of 0.8.

  The sixth environment features 10 communities. In this test environment, each community only uses 2 information services when they request information about a web service. With this low number of used information services, the information services in the system should be less requested, their truth trend is likely to increase during the simulation and, therefore, they should be less tempted to lie. The information services also start with a neutral initial truth trend of 0.0.

  Regarding payments and rewards, we set first the value of α at 5.0, β max at 20.0 and γ max at 40.0. The β and γ functions used in this environment are shown in Figures 6.16 and 6.17. The initial π is set to 20.0 and the π step to 5.0.

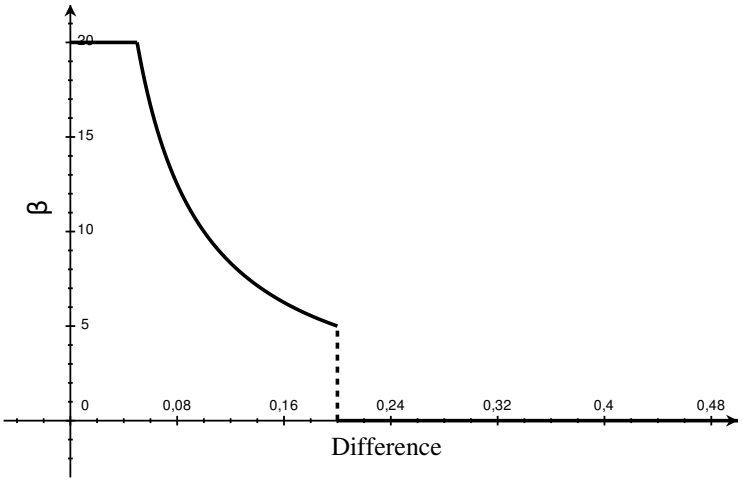

Figure 6.16: Environment 6 - β function

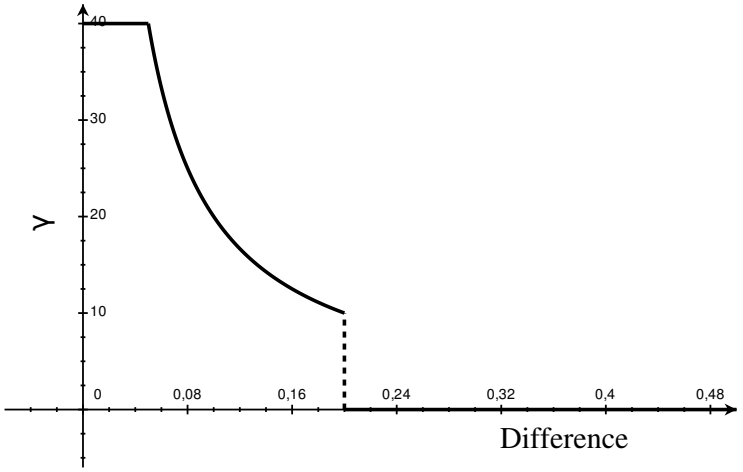

Figure 6.17: Environment 6 - γ function

The trust update function used with the environment is set as shown in Figure 6.18 with a max bonus value of 0.2 and a max malus of -0.1.

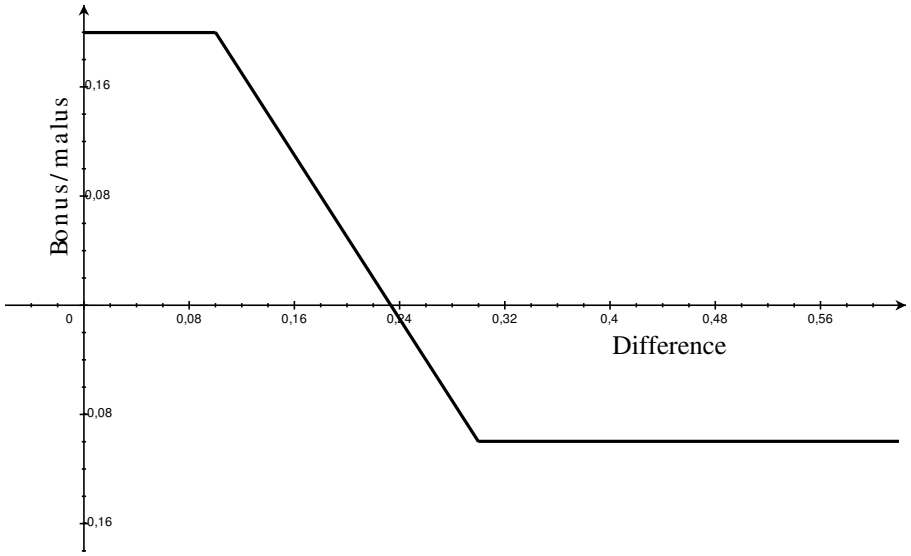

Figure 6.18: Environment 6 - Trust update function

- Environment 7 : Large number of communities with high knowledge of information services

  In this last test environment, we also used the second environment as basis but we introduced another modification about information services.

  We take a sample of 1000 web services as it provides results that allow good observations without having a too long simulation.

  The population of these web services is composed of only one category of services whose QoS ranges from 0 to 1 according to a random repartition and with a QoS variability of 0.8.

  The seventh environment features 30 communities that use 5 information services when they request information about a web service. Information services start with a neutral initial truth trend of 0.0. In this environment, the communities can also be aware of 4 times more information services than the number of used information services when requesting information about a web service. Usually, communities only know 2 times more information services.

  Regarding payments and rewards, we set first the value of α at 5.0, β max at 20.0 and γ max at 40.0. The β and γ functions used in this environment are shown in Figures 6.19 and 6.20. The initial π is set to 20.0 and the π step to 5.0.

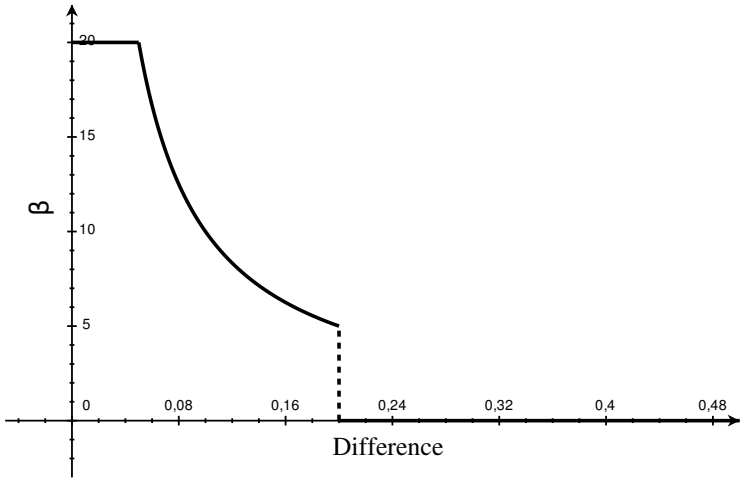

Figure 6.19: Environment 7 - β function

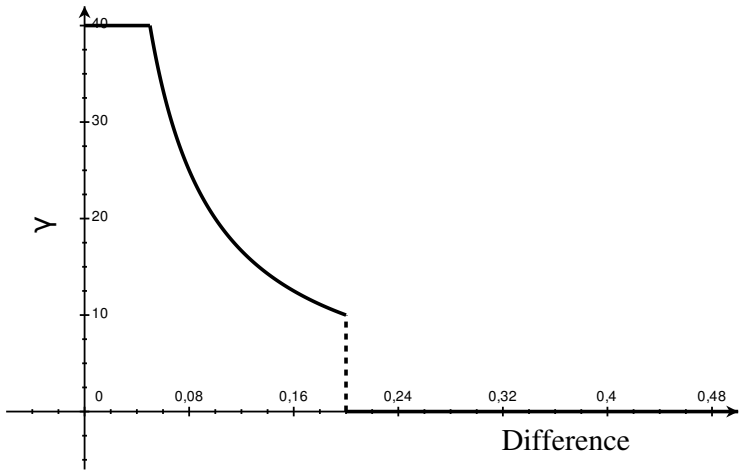

Figure 6.20: Environment 7 - γ function

The trust update function used with the environment is set as shown in Figure 6.21 with a max bonus value of 0.2 and a max malus of -0.1.

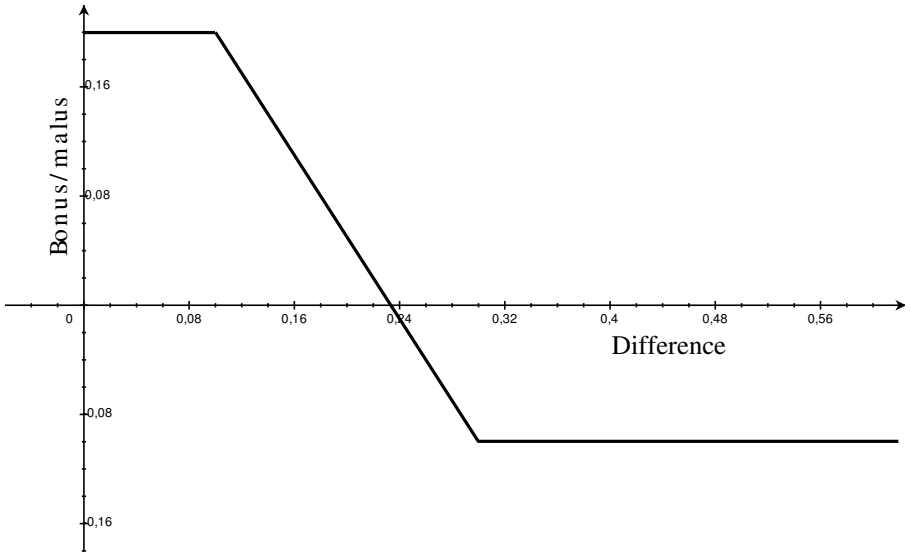

Figure 6.21: Environment 7 - Trust update function

6.2 Test environments synthesis

In Table 6.1, a summary of the parameters used for each test environment is presented.

	Test environment						
	1	2	3	4	5	6	7
Reward function used	No	Yes	Yes	Yes	Yes	Yes	Yes
Number of web services	1000	1000	1000	1000	1000	1000	1000
Number of categories of web services	1	1	1	1	1	1	1
QoS Variability	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Number of information services	10	10	10	10	10	10	10
Initial truth trend	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Number of communities	10	10	10	10	10	10	30
Number of used information services	5	5	5	5	5	2	5
Number of known information services	10	10	10	10	10	4	20
Minimum reputation value	0.5	0.5	0.5	0.5	0.5	0.5	0.5
α	5.0	5.0	5.0	5.0	5.0	5.0	5.0
β	20.0	20.0	20.0	20.0	20.0	20.0	20.0
γ	40.0	40.0	40.0	40.0	40.0	40.0	40.0
Initial π	20.0	20.0	40.0	40.0	40.0	20.0	20.0
π step	5.0	5.0	10.0	10.0	10.0	5.0	5.0
Maximum bonus	0.2	0.2	0.2	0.2	0.5	0.2	0.2
Maximum malus	-0.1	-0.1	-0.1	-0.5	-0.1	-0.1	-0.1

Table 6.1: Test environments parameters

6.3 Results analysis

In the previous section, we defined the several test environments that we used with our simulation to illustrate our theoretical solution. In this section, we analyse the results we obtained for each environment. This analysis is done according to a systematic approach.

For each environment, we first remind the main properties of the environment. We then examine the repartition of the web services in the communities at the end of the simulation. This graph is interesting because it gives a good idea whether or not the result of the simulation is satisfying.

Next, we take a look at the evolution of the trust average and truth trend at the end of the simulation of each of the information services. It should indicate how communities react toward information services' behaviour.

Finally, we consider the Truth/Lie ratio evolution. This ratio presents the global network action tendency. The evolution of the latter should reflect the effects of the chosen parameters on this tendency.

To conclude this section, we compare the results we obtained, see if they are consistent and verify the theoretical solution we proposed.

- Environment 1 : Basic environment without R

In this first test environment, we set up a population of 1000 web services distributed in one category whose QoS ranges from 0 to 1 and QoS variability equals 0.8. This environment also features 10 information services with an initial truth trend of 0, 10 communities using 5 information services and a minimum reputation value of 0.5.

The value of α is 5, β max 20, γ max 40, initial π 20 and π step 5. The maximum bonus is 0.2 and the maximum malus is -0.1.

The main properties of this first test environment is that the reward function is not used.

After the run of the simulation, the results we obtained with this environment are not really satisfying.

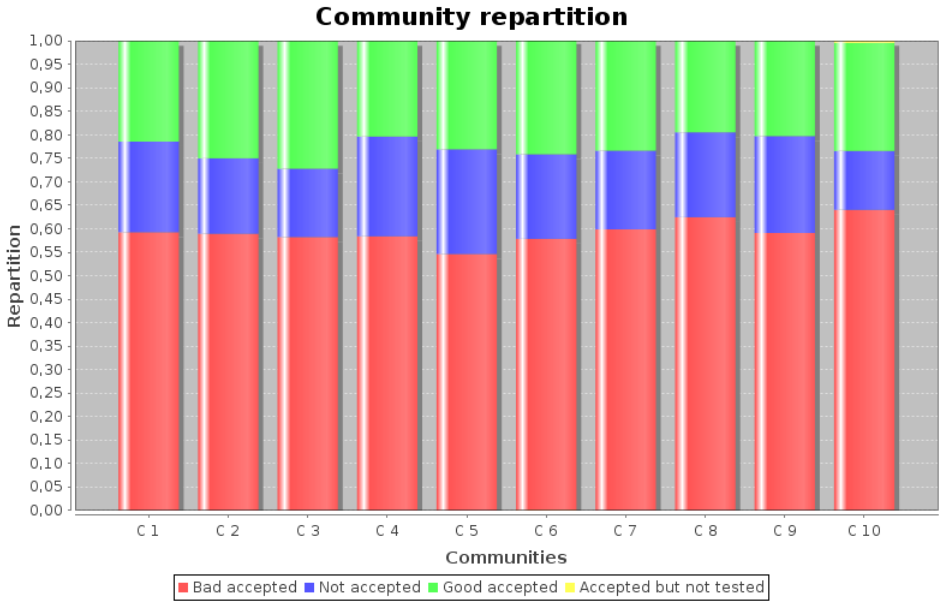

Figure 6.22: Environment 1 - Community repartition

As we can see in the previous Figure 6.22, the repartition of web services from each community at the end of the simulation is not very good. Indeed, the proportion indicates that lots of bad web services have been accepted. This can be explained by the fact that the maximum malus of the trust update function is very low and, therefore, information services are not

enough penalised when they choose to lie.

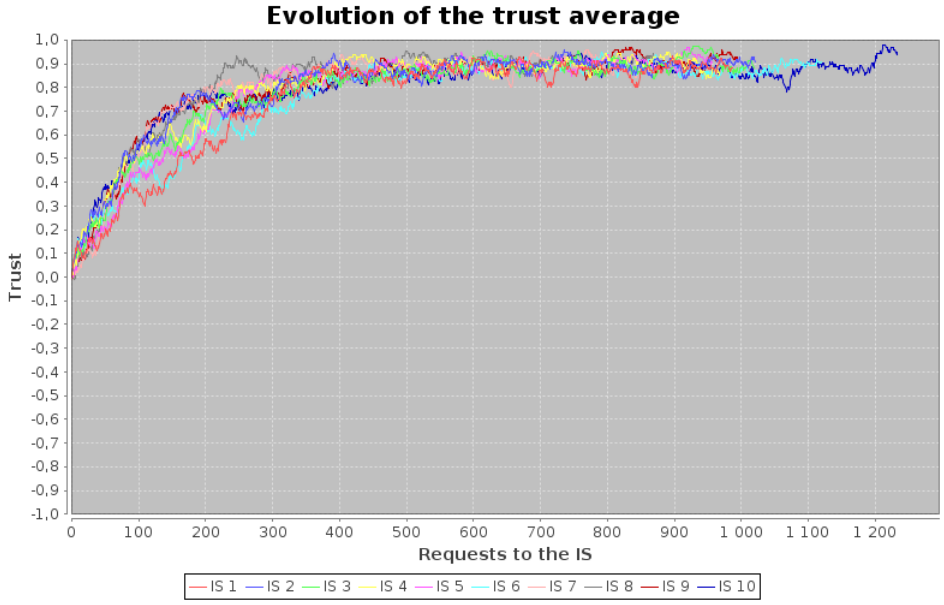

Figure 6.23: Environment 1 - Trust evolution

Furthermore, Figure 6.23 shows us that despite this bad repartition, the trust average of the different information services evolves positively during the simulation. We can tell that this situation is due to the difference between the maximum bonus and the maximum malus of the trust update function. Thanks to it, the trust increases more rapidly than it decreases.

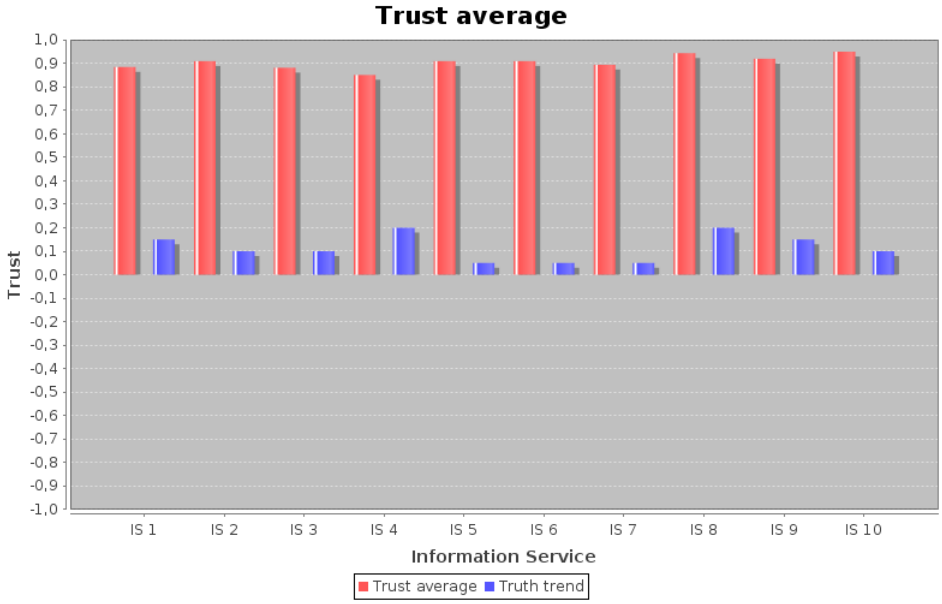

Figure 6.24: Environment 1 - Trust

As mentioned previously, Figure 6.24 indicates how high is the trust average of each information service at the end of this first simulation.

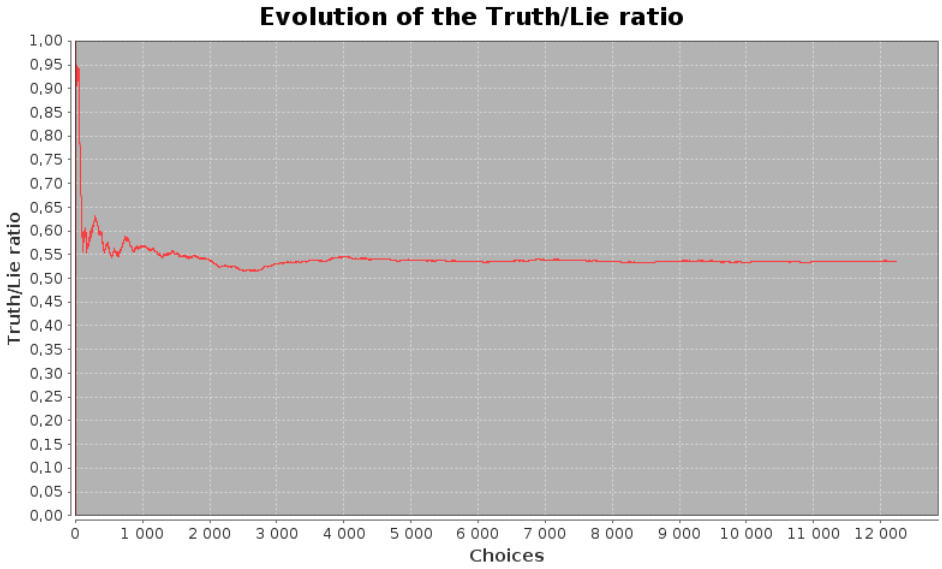

Figure 6.25: Environment 1 - Truth/Lie ratio

Finally, we can see in Figure 6.25 that the Truth/Lie ratio is stable.

- Environment 2 : Basic environment with R

In this second test environment, we set up a population of 1000 web services distributed in one category whose QoS ranges from 0 to 1 and QoS variability equals 0.8. This environment also features 10 information services with an initial truth trend of 0, 10 communities using 5 information services and a minimum reputation value of 0.5.

The value of α is 5, β max 20, γ max 40, initial π 20 and π step 5. The maximum bonus is 0.2 and the maximum malus is -0.1.

The difference between the first environment is that the reward function is now used and we should therefore see an improvement in the results.

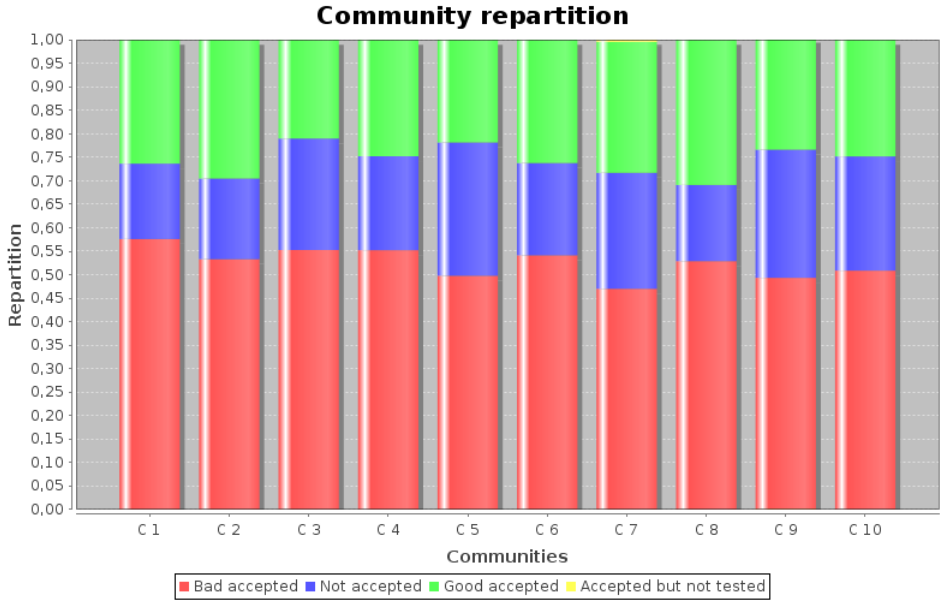

Figure 6.26: Environment 2 - Community repartition

As we can see in Figure 6.26, the repartition of web services in the communities at the end of the simulation has improved compared to the first environment. The proportion of accepted bad web services has decreased and is now at 52.3%. In the previous environment, this percentage was 58.9%. We can tell that the introduction of R had a positive but limited impact on the results of the repartition.

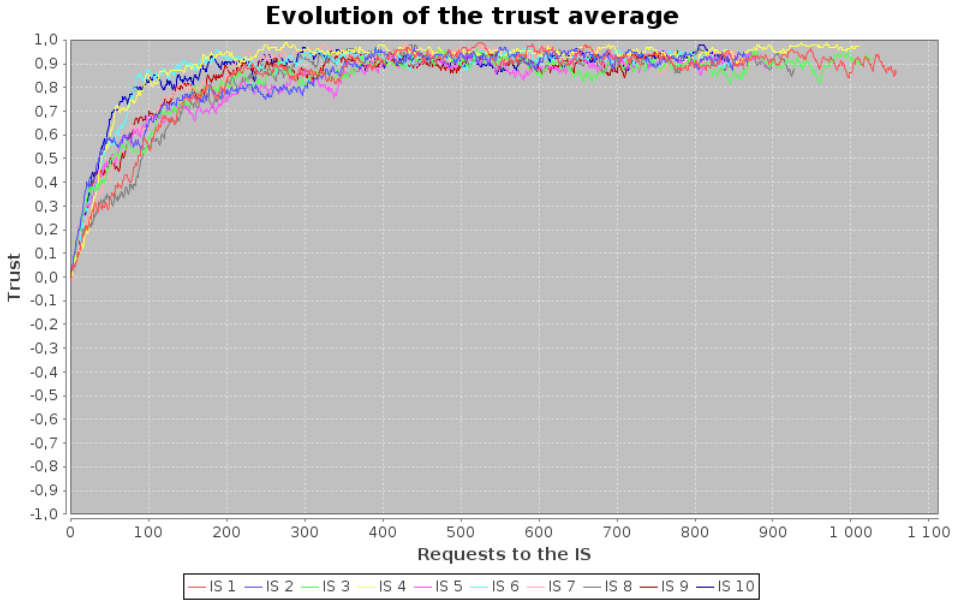

Figure 6.27: Environment 2 - Trust evolution

Regarding the evolution of the trust average of each information services, the conclusion is the same as for the first environment. Since we did not modify the trust update function, the results are logically very similar.

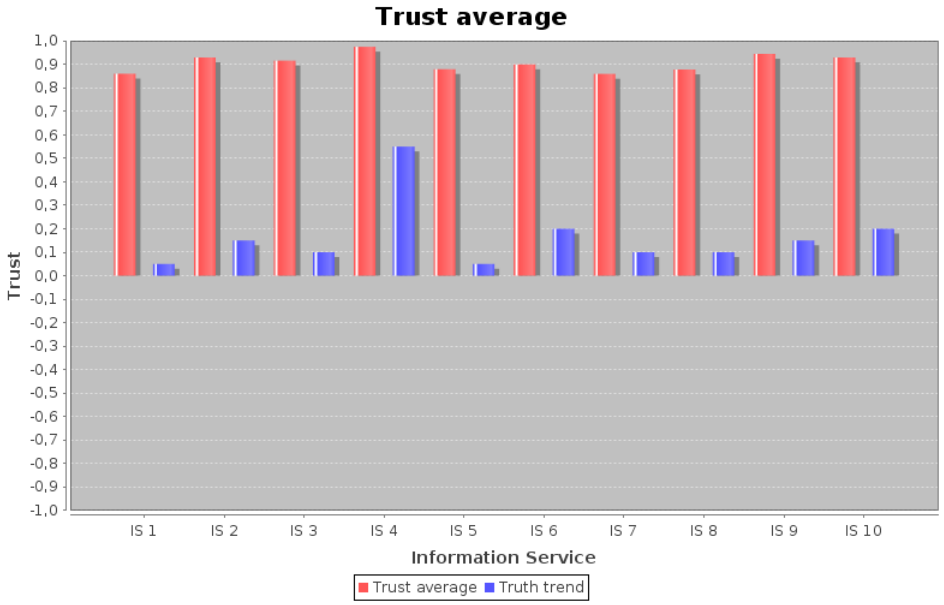

Figure 6.28: Environment 2 - Trust

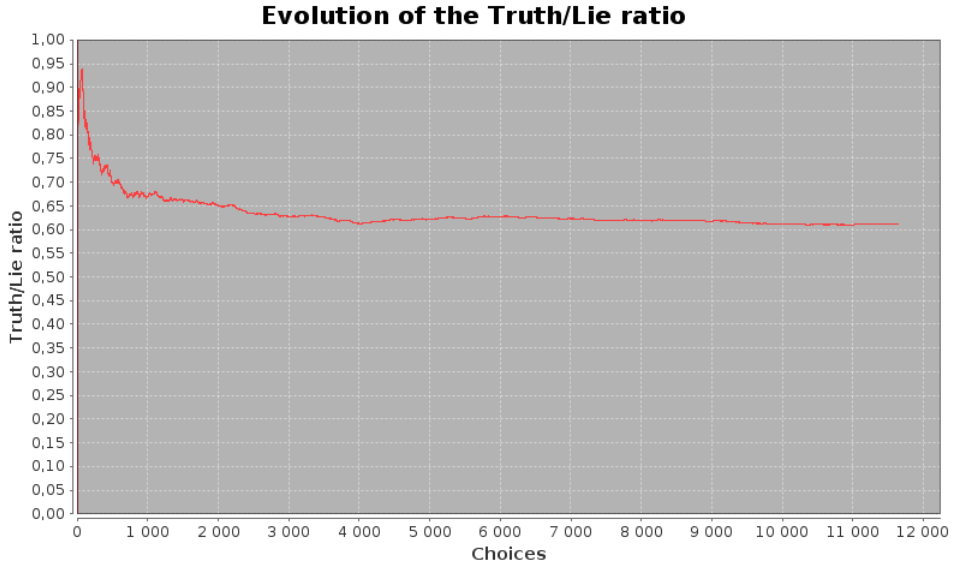

Figure 6.29: Environment 2- Truth/Lie ratio

- Environment 3 : Increased initial π and π step

In this third test environment, we set up a population of 1000 web services distributed in one category whose QoS ranges from 0 to 1 and QoS variability equals 0.8. This environment also features 10 information services with an initial truth trend of 0, 10 communities using 5 information services and a minimum reputation value of 0.5.

The value of α is 5, β max 20, γ max 40, initial π 40 and π step 10. The maximum bonus is 0.2 and the maximum malus is -0.1.

Compared to the second environment, we mainly modify and increase the value of π parameters. As it was expected, the results with this environment are even worse than previously.

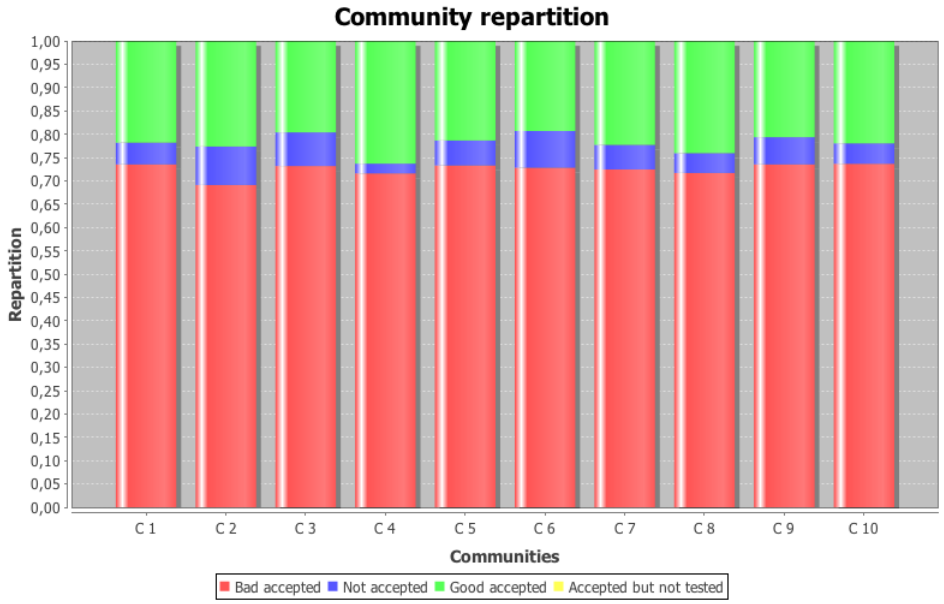

Figure 6.30: Environment 3 - Community repartition

Figure 6.30 clearly shows that the quality of the repartition of the web services in the communities has been decreased, with a lot more bad web services accepted within the communities by comparison with previous environments. This can easily be explained by the fact that information services are more tempted to lie thanks to the higher initial π and π step.

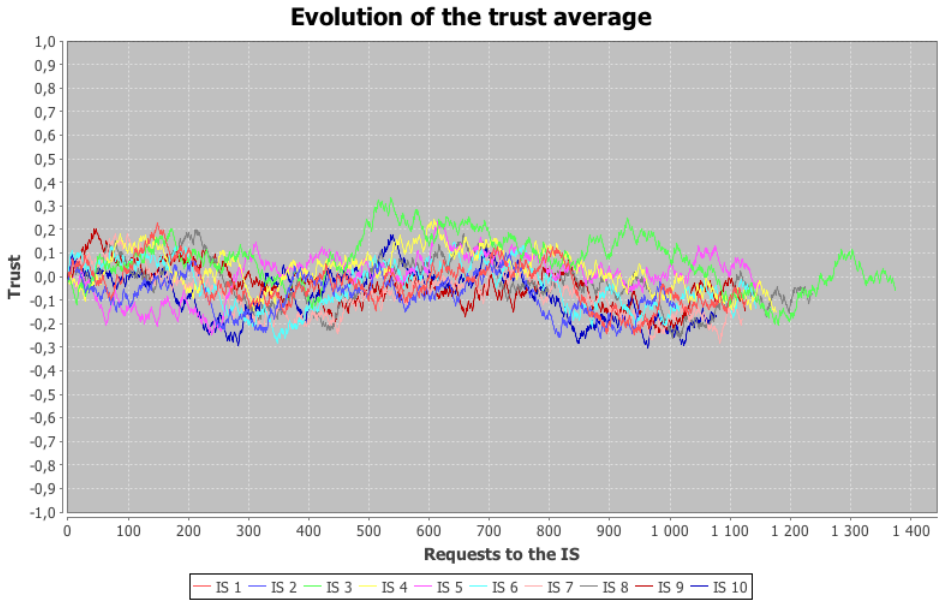

Figure 6.31: Environment 3 - Trust evolution

Logically, the evolution of the trust average is not as positive as in the second environment due to the higher number of bad web services that were accepted by the communities. Nevertheless, the difference between the maximum bonus and the maximum malus still prevents this evolution from being too negative. Indeed, we can see in Figure 6.31 that the trust average did not change far away from the neutral value of 0.0.

Compared to the previous environment, the Truth/Lie ratio (Figure 6.33) has been degraded. This result is quite logic. As we give the information services high initial π and π step, they lie a lot more, resulting in a lower Truth/Lie ratio.

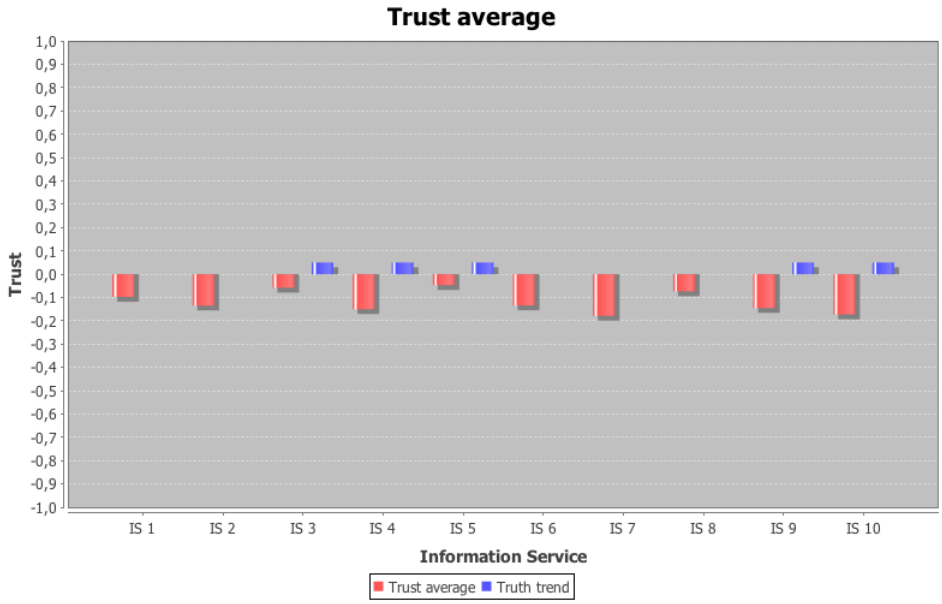

Figure 6.32: Environment 3 - Trust

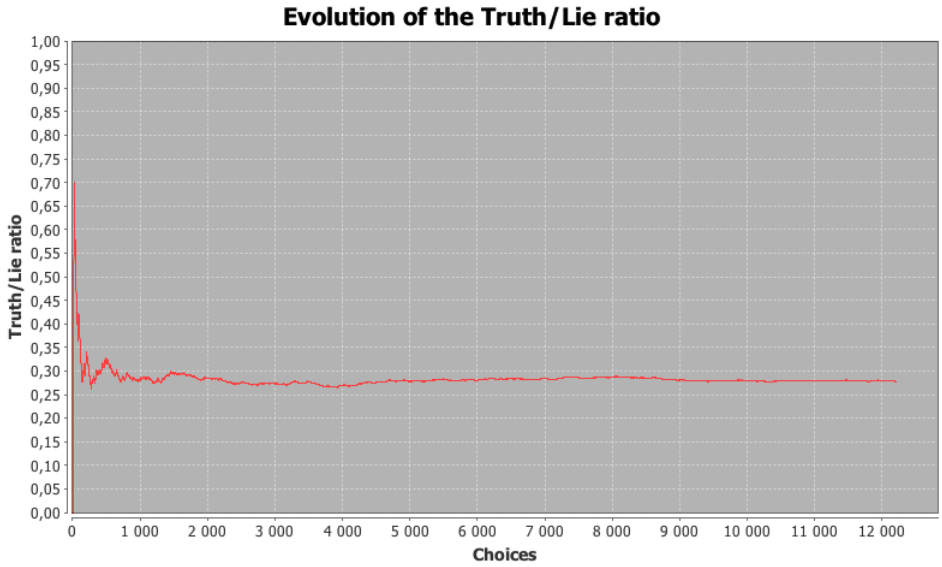

Figure 6.33: Environment 3 - Truth/Lie ratio

- Environment 4 : Increased maximum malus

In this fourth test environment, we set up a population of 1000 web services distributed in one category whose QoS ranges from 0 to 1 and QoS variability equals 0.8. This environment also features 10 information services with an initial truth trend of 0, 10 communities using 5 information services and a minimum reputation value of 0.5.

The value of α is 5, β max 20, γ max 40, initial π 40 and π step 10. The maximum bonus is 0.2 and the maximum malus is -0.5. By increasing the maximum malus from -0.1 to -0.5, the results of this simulation should have been substantially improved in comparison with what we obtained with the previous environment. Indeed, that is what we can observe with the following figures.

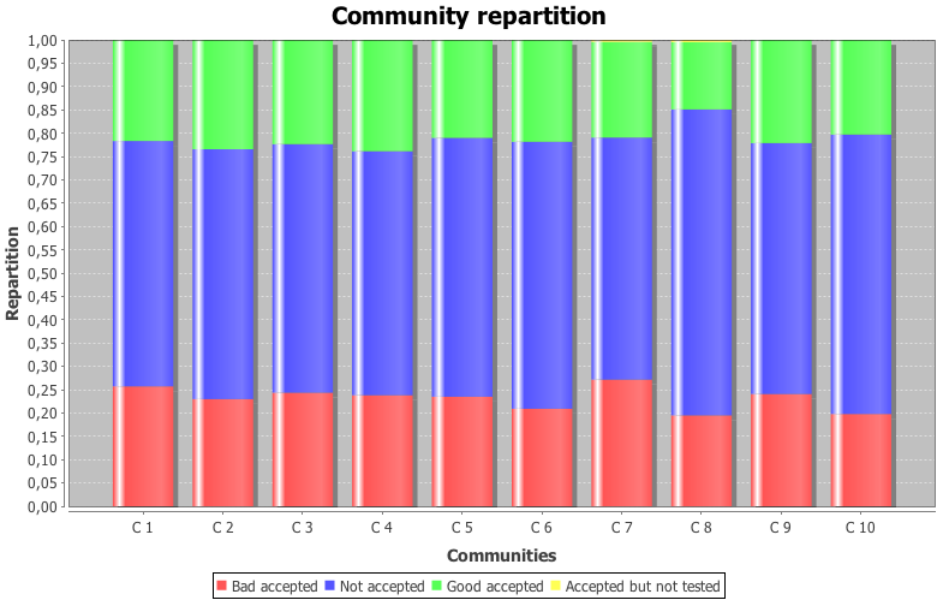

Figure 6.34: Environment 4 - Community repartition

First, let us analyse the Figure 6.34 and the repartition of the web services among the different communities at the end of the simulation. As we can see, the proportion of bad web services accepted by the communities is here far less important than in environment 3 (Figure 6.30). The percentage of bad web services accepted was 72,45% in the previous environment, it is now decreased to 23,19%. The explanation is that, thanks to an higher maximum malus than the one used in environment 3, information services are less tempted to lie and are severely punished if they do. Therefore, a lot

more web services are rejected by the communities, as shown in the Figure 6.34 by the central blue portions.

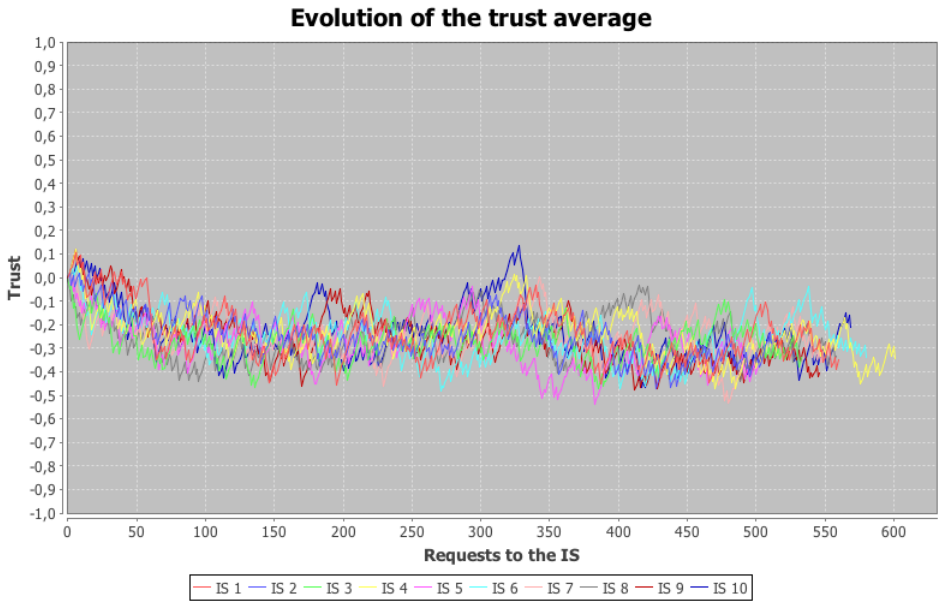

Figure 6.35: Environment 4 - Trust evolution

Despite the satisfying repartition of web services in the communities, the evolution of the average trust of each information service is not positive (Figure [6.35). At the end of the simulation, none of the information services has a positive trust average. We can explain this situation by the fact that the maximum malus is very important. Indeed, even though a lot of bad web services are refused by communities, indicating that the information services are mostly telling the truth, the important malus has a huge impact on the trust. This impact can only partially be compensated by the lower maximum bonus.

At the end of the simulation, we can see in Figure 6.36 that the truth trend of some of information services has been increased a little bit.

In this environment, information services had a clear incentive to tell the truth. This is shown by the Figure 6.37 where the Truth/Lie ratio is high. It also tends to slightly increase during the simulation.

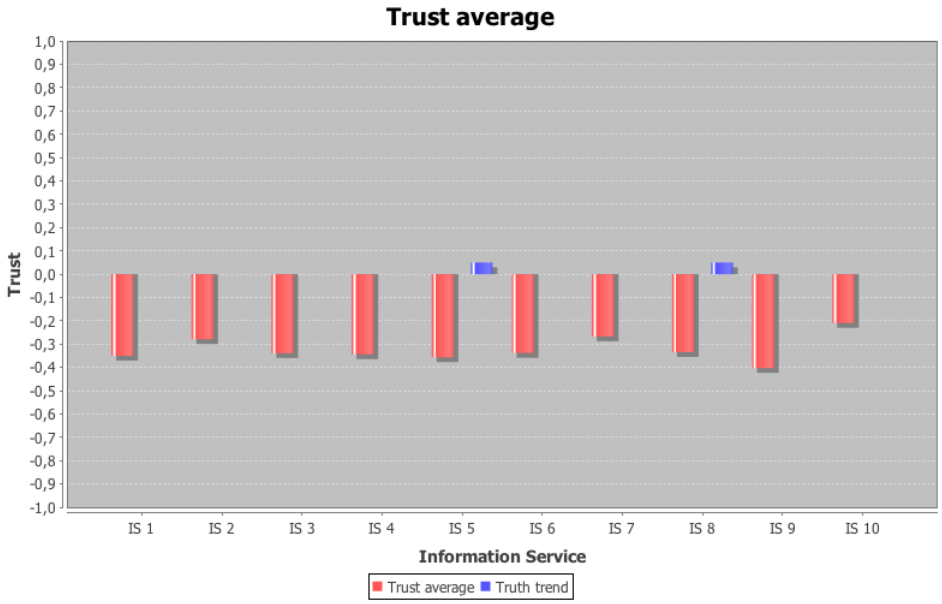

Figure 6.36: Environment 4 - Trust

Figure 6.37: Environment 4 - Truth/Lie ratio

- Environment 5 : Increased maximum bonus

In this fifth test environment, we set up a population of 1000 web services distributed in one category whose QoS ranges from 0 to 1 and QoS variability equals 0.8. This environment also features 10 information services with an initial truth trend of 0, 10 communities using 5 information services and a minimum reputation value of 0.5.

The value of α is 5, β max 20, γ max 40, initial π 40 and π step 10. The maximum bonus is 0.5 and the maximum malus is -0.1.

In this environment, we take inspiration from the third environment but we modify it in the opposite way compared to the environment 4. Instead of increasing the maximum malus, we boosted the maximum bonus. Unlike the previous environment, the results here are totally unsatisfying as we see in the next figures.

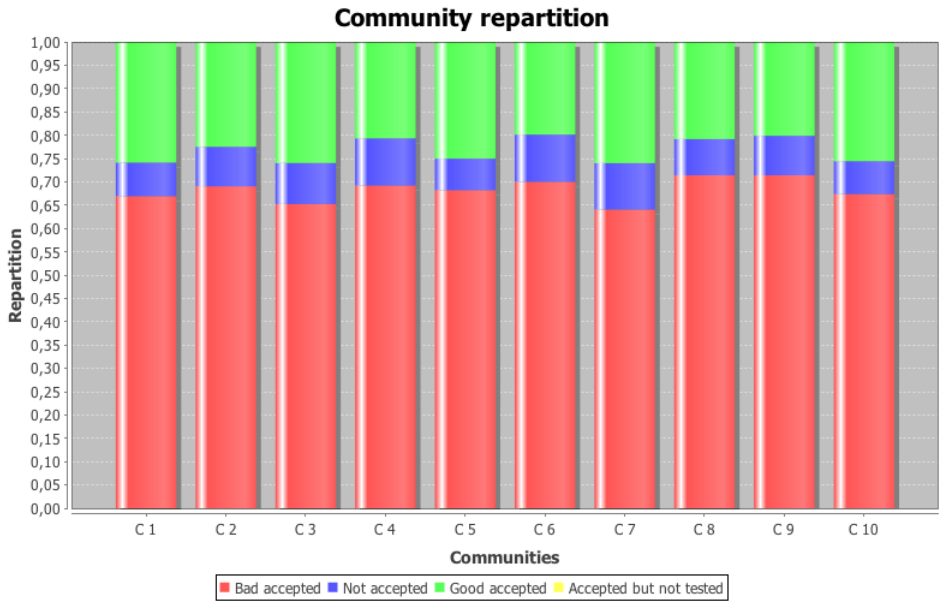

Figure 6.38: Environment 5 - Community repartition

First, let us begin with the repartition of the web services in the communities presented in the Figure 6.38. The results are comparable with those obtained with the second environment. The proportion of bad web services that are accepted is very high. In average, the percentage of those web services is 68,31% which is close to what we noticed in the third environment

(72,45%).

Such an important percentage of bad web services accepted can be explained by the combination of the high value of the π parameters and the low maximum malus. Indeed, the information services are tempted to lie thanks to the substantial incentives and are not discouraged because of the low maximum malus.

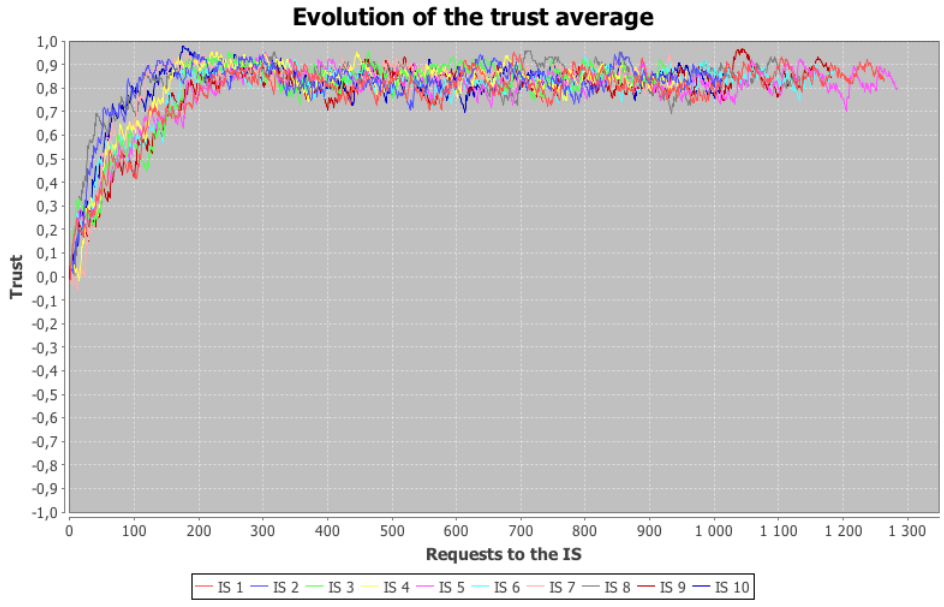

Figure 6.39: Environment 5 - Trust evolution

Despite the consequent proportion of bad web services accepted, Figure 6.39 indicates how positively the trust average of all the information services evolves during the simulation. Such a result is not surprising. The high maximum bonus of the trust update function is responsible for the quick compensation of the loss generated by one or several lies by a considerable gain when telling the truth only once.

As for the repartition of web services in the communities, the Truth/Lie ratio in this environment is comparable to what we obtained in the third environment, even if the result is slightly better.

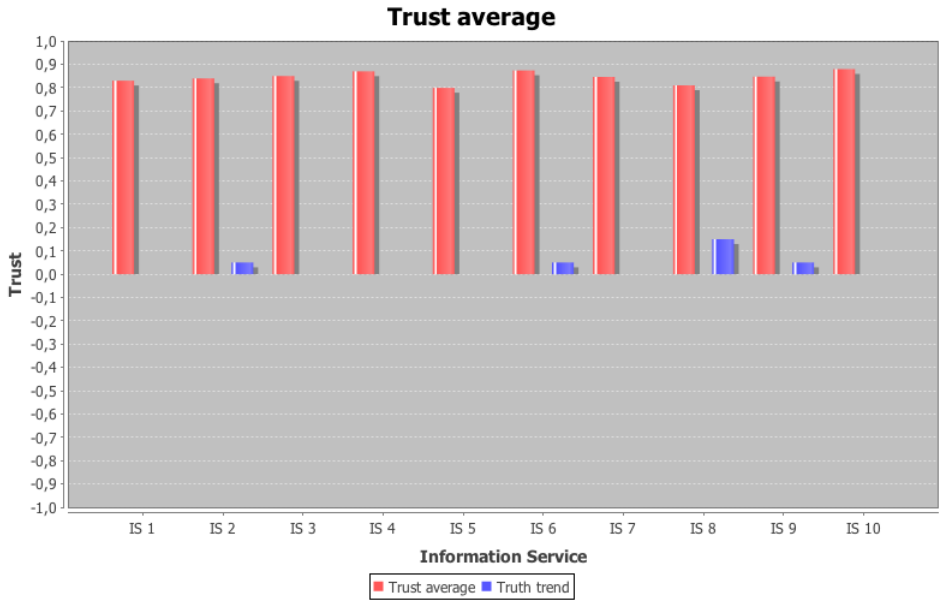

Figure 6.40: Environment 5 - Trust

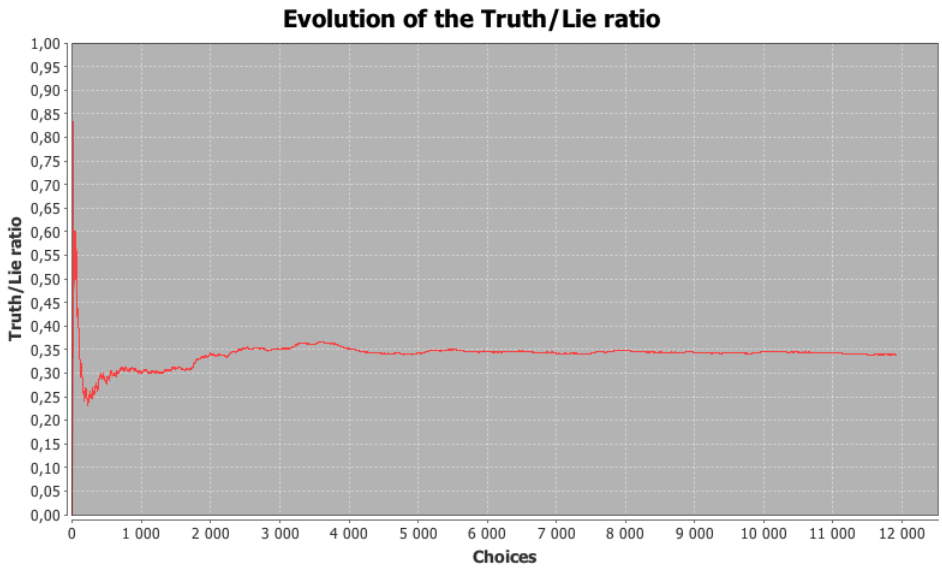

Figure 6.41: Environment 5 - Truth/Lie ratio

- Environment 6 : Low number of used information services

In this sixth test environment, we set up a population of 1000 web services distributed in one category whose QoS ranges from 0 to 1 and QoS variability equals 0.8. This environment also features 10 information services with an initial truth trend of 0, 10 communities using 2 information services and a minimum reputation value of 0.5.

The value of α is 5, β max 20, γ max 40, initial π 20 and π step 5. The maximum bonus is 0.2 and the maximum malus is -0.1.

To set up this environment, we go back to the parameters of the second environment. The main change we decide to introduce is to reduce from 5 to 2 the number of information services used by each community when asking for information about a particular web service.

By reducing the number of information services used by the communities, the probability for an information service to be used has decreased. Therefore, the truth trend of the information services is more likely to increase and, as a result, we see an improvement of the outputs as the following figures indicate.

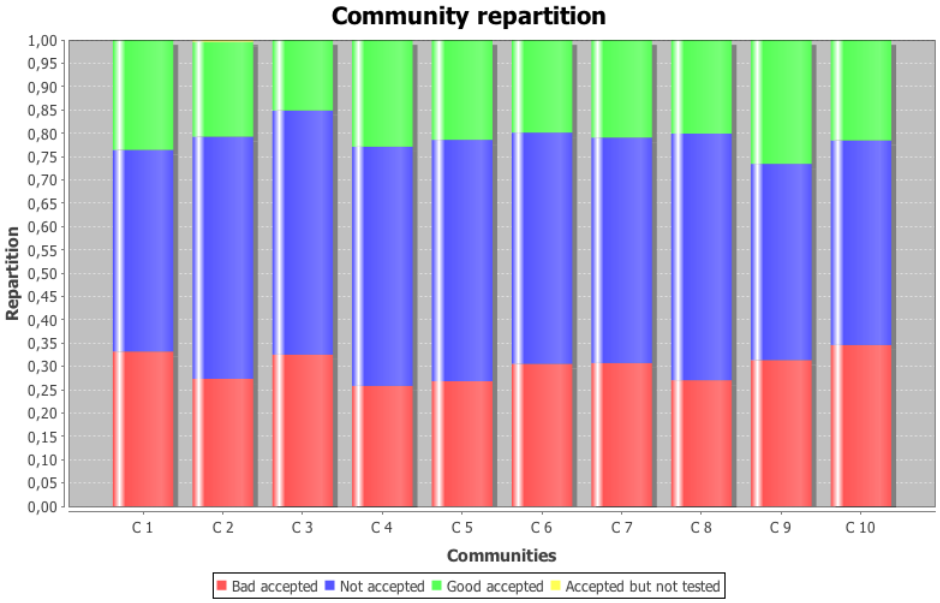

Figure 6.42: Environment 6- Community repartition

When we look at the repartition of web services in the communities at the end of the simulation in the Figure 6.42, we can see that the results are better than what we obtained with the second environment. In the latter, the proportion of bad web services that were accepted by the communities was 52.3% while here it is 30.1%. The progression of the truth trends can explain why the information services chose to tell the truth.

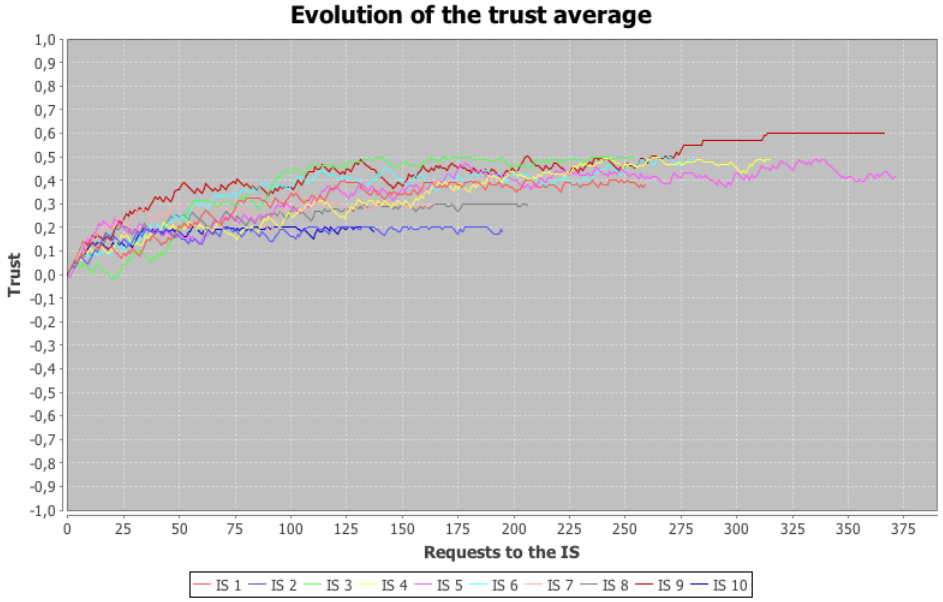

Figure 6.43: Environment 6 - Trust evolution

Figure 6.43 shows that the average trust of all the information services evolves in a positive and continuous way during this simulation. As expected, we can see in Figure 6.44 that the truth trend for each information service at the end of the simulation can be very high.

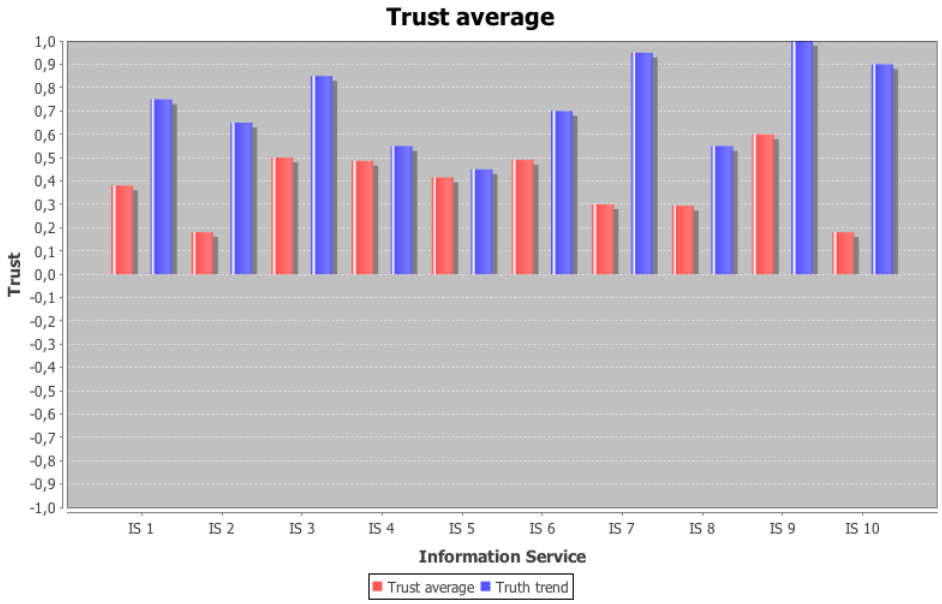

Figure 6.44: Environment 6 - Trust

Looking at the Truth/Lie ratio in Figure 6.45, we can tell that results are good. Indeed, the ratio gradually increases during the simulation which indicates that information services are more likely to tell the truth in the future interactions.

Figure 6.45: Environment 6 - Truth/Lie ratio

- Environment 7 : Large number of communities with high knowledge of information services

  In this last test environment, we set up a population of 1000 web services distributed in one category whose QoS ranges from 0 to 1 and QoS variability equals 0.8. This environment also features 10 information services with initial truth trends of 0 and 30 communities using 5 information services and a minimum reputation value of 0.5.

  The value of α is 5, β max 20, γ max 40, initial π 20 and π step 5. The maximum bonus is 0.2 and the maximum malus is -0.1.

  In this last environment, we increase the number of information services in the system and set it to 30. Communities are also aware of 4 times more information services than the number of information services they use when they request information about a web services.

  Thanks to these modifications, the probability for an information service to be chosen should decrease and, therefore, its truth trend is likely to increase. As we can see in the following figures, this give us some good results.

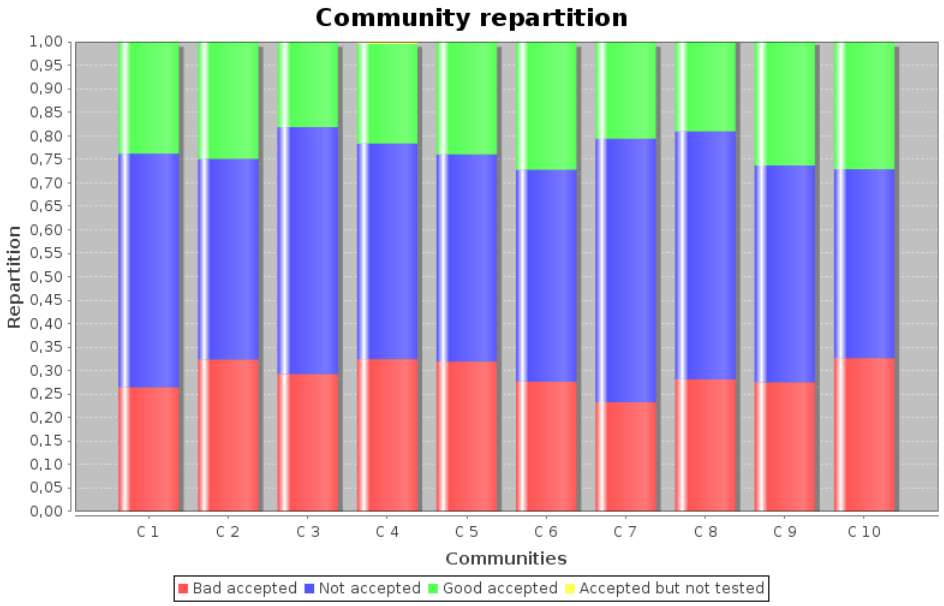

Figure 6.46: Environment 7 - Community repartition

First, let us analyse the repartition of the web services in each community at the end of the simulation. As we can see in Figure 6.46, this repartition

is good, with a low percentage of bad web services accepted.

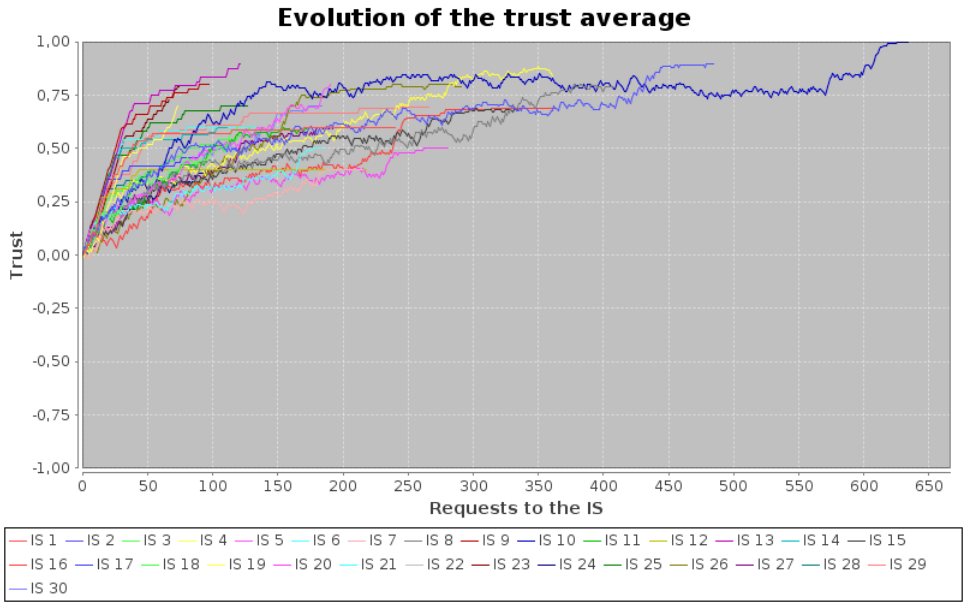

Figure 6.47: Environment 7 - Trust evolution

Figure 6.47 indicates that the evolution of the trust average of each information service has positively evolved during the the simulation.

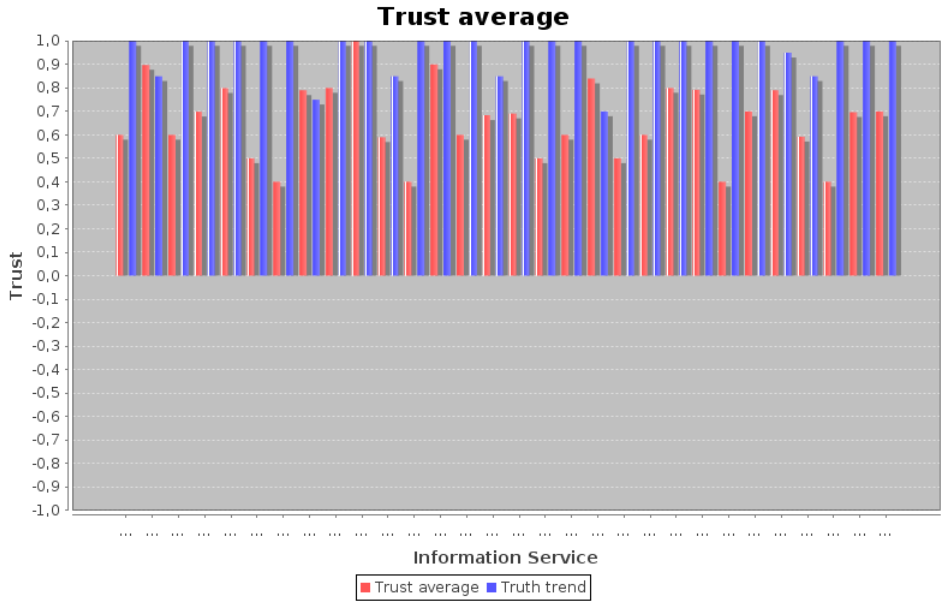

Figure 6.48: Environment 7 - Trust

We can see in Figure 6.48 that, as expected, the truth trend of all the information services at the end of the simulation is very high, even maximum for the majority of the information services.

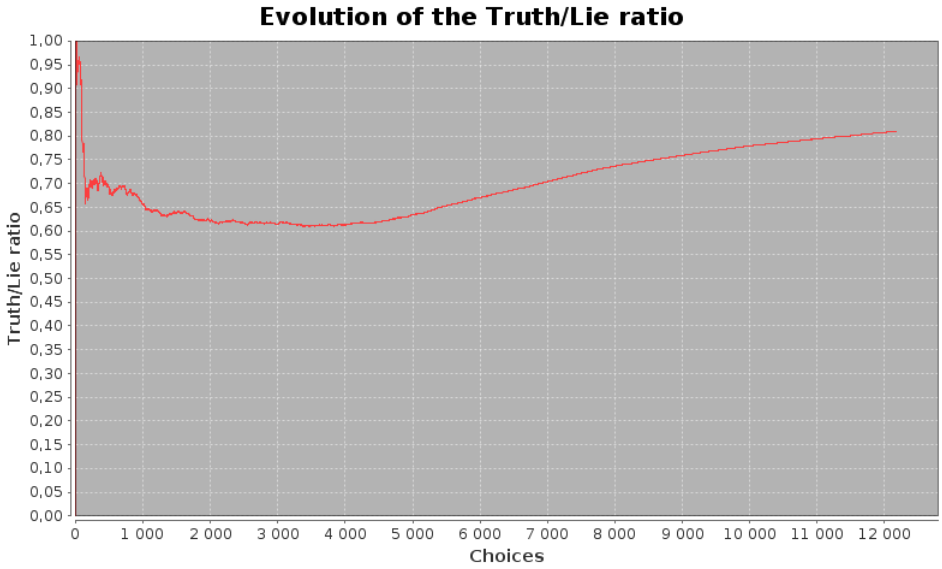

Figure 6.49: Environment 7 - Truth/Lie ratio

Similarly to the previous environment, the Truth/Lie ratio shown in Figure 6.49 is high and keeps increasing during the simulation. The final value in this environment is even better than what we obtained in the sixth environment. This is mainly due to the fact that, over time, the trend of the information services to tell the truth increases.

6.4 Conclusion

In this section, we propose a synthesis of the results we obtained through the simulation of the test environments we presented in Section 6.1. We first compare the quality of the outputs of each test environment. We use for that purpose graphs that combine the main properties of the test environments.

Then we end this section by making a conclusion that summarises the main ideas and observations we made during this experimentation phase.

6.4.1 Results comparison

To begin this results comparison, we are interested in the repartition of the web services in the communities at the end of each simulation.

As we can see in Figure 6.50, we have two distinct groups of repartition. Indeed, the test environments 1, 2, 3 and 5 provide repartition with a high density of bad web services accepted in the communities. Conversely, with the test environments 4, 6 and 7, this density is lower and we observe that a lot more bad web services have been rejected by the communities.

From these observations, we can draw some conclusions. First of all, modifying the value of the maximum malus proves to be more effective than modifying the maximum bonus. This is due to the fact that with an important malus, information services are severely punished when lying to a community and are therefore more tempted to tell the truth.

Environments 6 and 7 show us good results as bad web services accepted are approximately twice less numerous than in environments 1, 2, 3 and 5, and as bad web services rejected are at least twice more numerous than in those environments. Those two environments have in common the fact that, in both of them, an information service has less chance of being chosen.

In environment 6, this is due to the fact that a community only uses 2 information services when asking for information about a web service, and only knows 4 of them. In environment 1, 2, 3 and 5, an information service uses 5 information services and knows 10.

In environment 7, this phenomenon is explained by the fact that every community knows four times more information services than it uses, whereas in the other environments, a community only knows twice more information services

than it uses.

Let's illustrate this by those 3 numbers, showing for each environment the probability for an information service to be chosen at the "first round" by at least 1 community :

Environment	Probability
1 to 5	0,999
6	0,737
7	0,838

Table 6.2: Probability of information service "first round" choice per environment

If an information service is not chosen in a specific period of time, the truth trend of the information service grows and the probability for this information service to be honest in the future is higher. This explains why environments 6 and 7 have better results in this repartition.

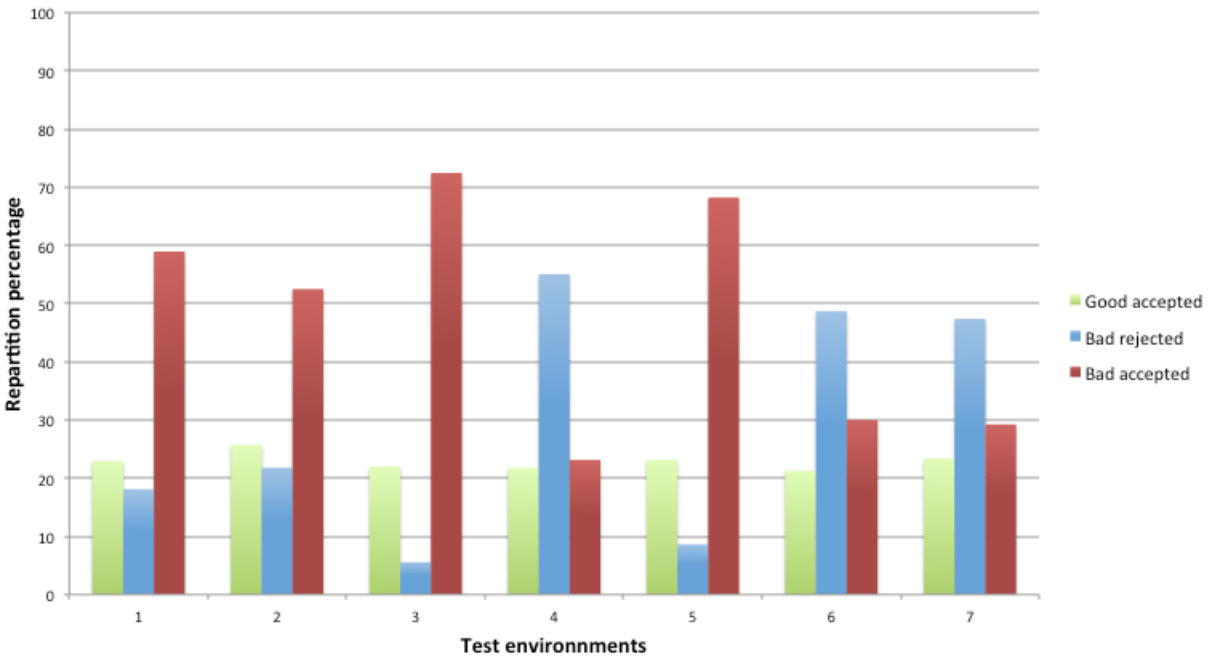

Figure 6.50: Communities repartition

In Figure 6.51, we compare the trust average that all communities have towards all information services in each test environment. In this graph, we can see

that we obtain a negative value in two cases, in test environments 3 and 4. With regard to the third test environment, we increase the value of the initial π and the π step. With these modifications, the information services are more tempted to lie, resulting in a low final trust average.

As for the fourth test environment, such a low trust average can seem surprising at first sight since we observe a good community repartition in Figure 6.50. This is explained by the fact that the maximum malus set in this case is very important. Information services are therefore severely punished each time they lie.

If we look at the trust average of the fifth test environment, which also features modified π parameters values, we can see that the important maximum bonus has a opposite effect and that the resulting trust average is high.

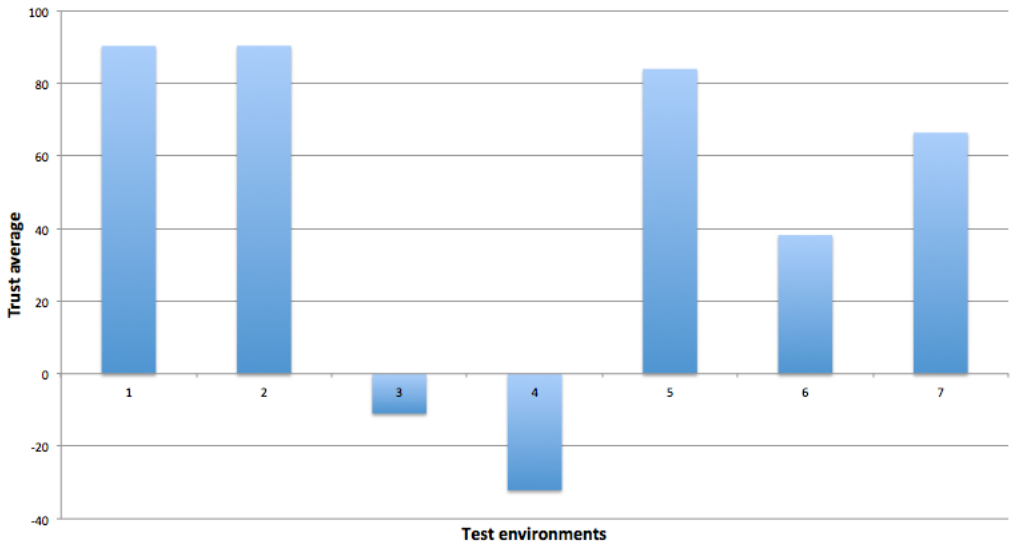

Figure 6.51: Trust average

We then analyse the trust trend average of all information services for each test environment. We observe in Figure 6.52 that the truth trend is very high in the case of the two last test environments. In the latter, we modify the parameters of the simulation so that communities use less information services when requesting about the quality of a web service (in the test environment 6) or have knowledge of more information services (in the test environment 7). As a result, information services in both test environments are less likely to be requested and their truth trend increases. It is consistent with the fact that in both situations, community repartitions are good.

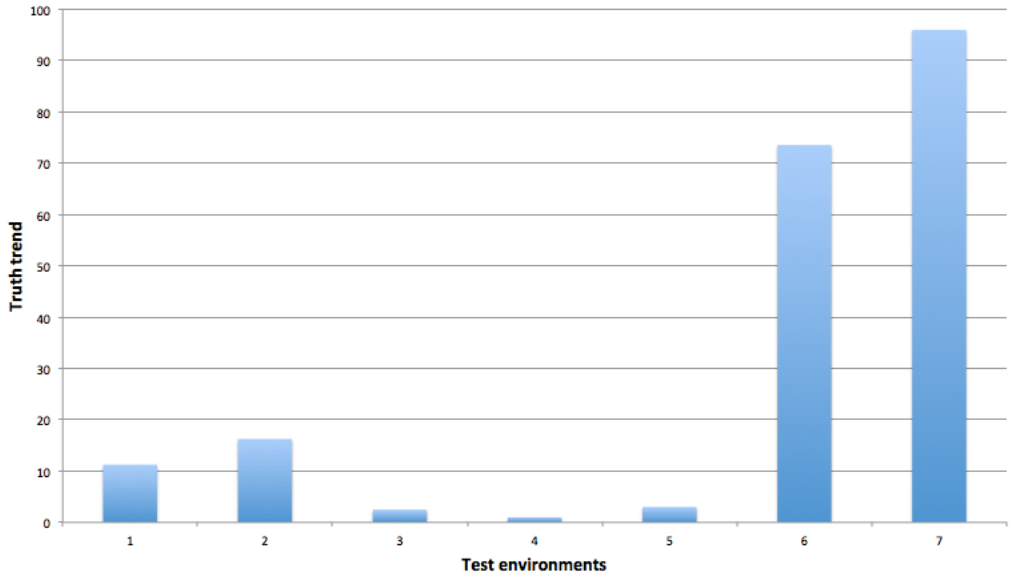

Figure 6.52: Truth trend

Finally, we take a look at the evolution of the Truth/Lie ratio of each of the test configuration.

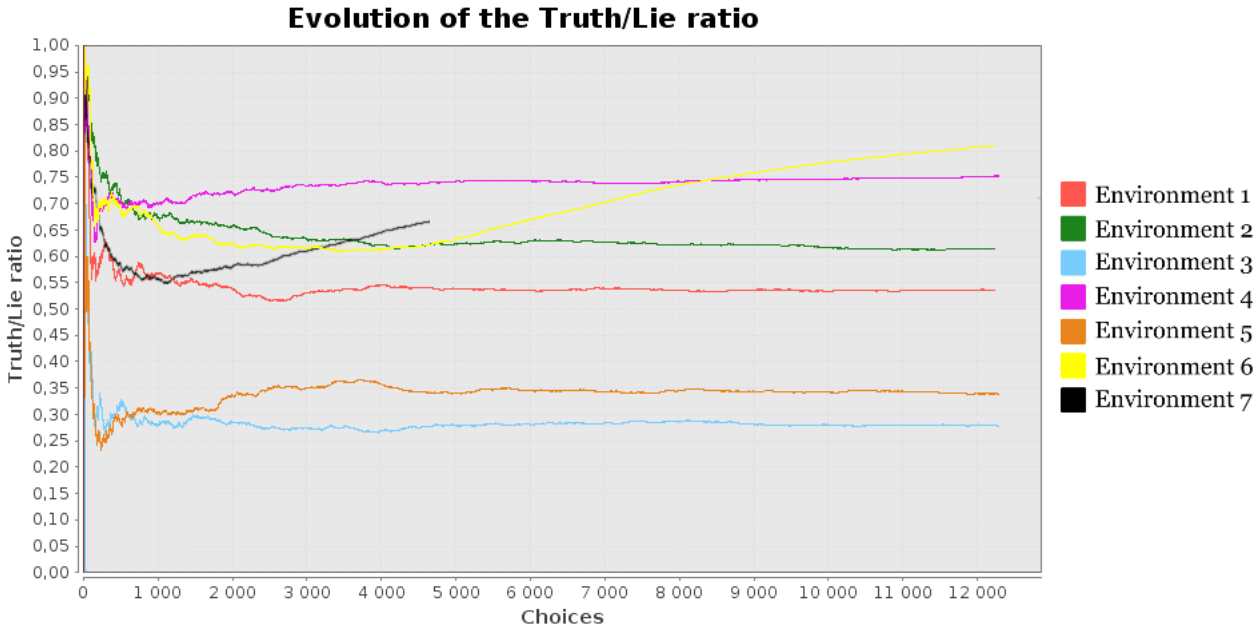

Figure 6.53: Truth/Lie ratio

We can see that the five first environments have a Truth/Lie ratio that varies

at the beginning, but quickly stabilises after 2000 or 3000 evaluations. The different level at which they stabilise are easily explained with the help of Figure 6.54[1]. Logically, environment that have a bigger proportion of bad web services accepted are the environment where information services lie the most, and the other way round.

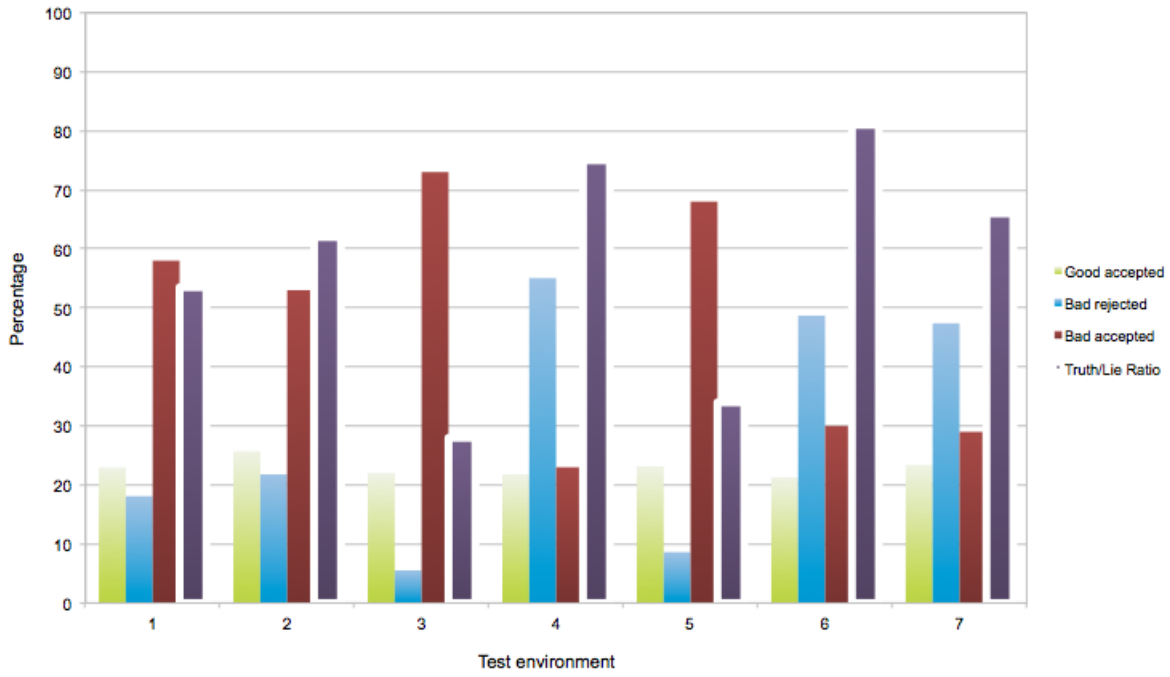

Figure 6.54: Truth/Lie ratio and community repartition comparison

Regarding the test environments 6 and 7, we saw in Figure 6.52 that the information services' truth trends are high at the end of the simulation. These high truth trends explain the increasing shape of the Truth/Lie ratio of these two test environments as information services are more likely to progressively tell the truth.

[1]Note that the value of Truth/Lie ratio in Figure 6.54 represents the final value of this ratio for every environment.

6.4.2 Final observations

Thanks to those simulation results, we managed to extract some guidelines in order to create environments that increase the probability for information services to be honest.

First, we can argue that the introduction of the reward function is not a solution per se, as we saw that the impact on the results were better but only in a slight way. The choice of a right environment is thus clearly decisive.

Concerning the parameters of the reward function, it is not its direct application that causes great modifications. However, its combination with an appropriate choice of values for max bonus and max malus has direct impacts on results. Increasing the max malus provides more honest answers from information services as lying becomes very penalising. On the other hand, increasing the max bonus does not appear to be very efficient as it increases the flaw identified in the results of the simulation of the second environment, that is to say that information services are more trusted than they should be.

Another way for communities to maximise their chance to get accurate information from information services is to get linked to the bigger possible number of information services. Doing so, a community has more choice when looking for information about a particular web service. As a logical consequence, the probability for an information service to be requested decreases. When having less choice, a community could be reduced to the obligation of selecting an information service which trust is not satisfying. In the situation where a community knows many information services, this kind of limited choice is avoided. Information services are put aside when lying and therefore gain motivation to tell the truth for future requests.

7 Conclusion

In this chapter, we recall the most important points of our investigation and summarise the problem we studied and the solution we introduced.

We have seen that a web service could try to join a community when it does not receive enough requests from users or when it is overloaded because it could threaten its survival. In order to accept this web service, the community has to perform an analysis of the latter to be sure that it can live up to the community's expectations in terms of reputation. Unfortunately, the community cannot rely on the information provided by the web service itself as it can produce fake information to improve its chance to be accepted. Therefore, we introduced a new type of service called information services.

The purpose of an information service is to gather and sell information about other web services' reputation. When a web service asks to join, a community sends queries to several information services about the latter and, based on the answers, decides to accept or reject the web service.

Unfortunately, the honesty of the information services cannot always be guaranteed. Indeed, we can imagine that a web service with a very bad reputation can try to corrupt the information services by giving them incentives to lie. The information services would then provide fake results to the communities and the web service would be accepted.

At this point, an information service can therefore be paid according to a 4-step mechanism. Three of the payments an information service can receive come from the communities while the last one is given by the web services. One of these payments (β) and the decision of accepting a web service is computed according to the average reputation value. Therefore, an information service has to take into account other information services' potential action before deciding

what to do. An information service achieves this reflection using game theory principles.

We used these game theory principles to study the possible behaviour of information services during their interactions with a community. We noticed that with this current payment mechanism, information services were heavily influenced by the incentive given by bad web services to make them fake the information they provide. Therefore, we had to define a payment mechanism where decisions taken by an information service have influence on its future payments. With this mechanism, telling the truth should bring in a big reward whereas lying should be punished.

If information services are offered big rewards to lie, it is more than likely that lying is what they decide to do. The problem for those information services is that, even if direct rewards are important, future rewards are likely to decrease. Indeed, communities notice when dishonest information services lie and request these information services less frequently. Information services receive less requests and therefore less rewards for information.
When deciding whether to lie or tell the truth, information services have to think about direct rewards but also future potential ones. We expressed this consideration in a combination of both rewards : the *reward function*. The reward function needs to balance direct rewards with the probability of being requested in the future. This probability is directly linked to the trust a community has towards an information service, this is why we used this value of trust as parameter to balance rewards.

In order to assess our model, we implemented a simulation tool. In this tool, we simulated the behaviour of the different entities (web services, information services and communities of web services). Each of these can be configured. This allowed us to set up different environments and scenarii. The aim of these simulations was to compare the outcome of the use of our model in these different environments. The execution of those several scenarii showed a great variability in the results, emphasizing the importance of the environment parameters.

Simulations helped us to underline two key elements that guarantee a certain honesty from information services.
First, information services tend to be more afraid by a big penalty than they are motivated by a big reward. Therefore, good results can be obtained for communities if they strongly decrease the trust they have towards an information service when the latter lies.
The other key element for an honest system is for communities to broaden their contact set. If a community knows a large amount of information services, it can choose to request only the more trusted ones. If the amount of information

services a community knows is big enough then lying only once is sufficient for an information service to never be chosen again. Information services are then more urged to tell the truth.

8 Limitations and future works

We are aware that the work we have accomplished has its own limitations. In some situations, it appears that flaws can be found in our theory. Furthermore, we sometimes postulate assumptions that seemed to create a truncated vision of the reality. In this last chapter, we discuss some of the weaknesses that could be subjects for future works in order to be corrected.

The first problem concerns the potential issue that can occur with the bribe web services offers. In the theoretical solution we developed, we try to ensure that the agents in the system adopt a honest behaviour by proposing a mechanism relying on 3+1 incentives. This mechanism combined with the evolution of the trust that the communities have towards the information services should counter the bribes of the web services. In our study, we assume that the value of these bribes are set to a reasonable scale compared to the payments received from the communities. Unfortunately, we could imagine situations where such incentives are very high. In that case, it is possible that the information services are really motivated to lie. This phenomenon might be increased if the game between the agents is played a few number of times, as explained in Section 4.7.2. Such a situation could be met, for example, in an environment where agents are in a stable state. In this environment, the amount of requests sent by users is manageable by the web services, so that web services are neither overloaded nor idle. They do not need to join a community in order to gain or distribute requests. As for the communities, the web services composing them is suitable to handle the requests and therefore do not feel the need to increase their size.

Another limitation we can point concerns the way payments are distributed. In our model, we consider that all agents are trustworthy regarding the payments they give. Indeed, when an information service has to be rewarded in exchange for its service or receive an incentive to lie, we assume that the communities and the web services are honest and actually pay it. Of course, it is possible that in

some cases, we could be dealing with dishonest agents that could promise high payments or incentives but might not be able to afford such amounts. In the future, this problem could be approached in order to make our model closer to the reality. We can imagine the introduction of some mechanisms that guarantee the honesty of the agents thanks to penalties that would be applied when a community or a web service refuses to make the payment.

The collusion between the agents of the system is another problem that we did not handle in our solution but that could raise some interesting questions. In our theoretical development, we assume that no interaction exists between the agents apart from the requests of service and information. Unfortunately, it could sometimes well be the case. For example, a group of several information services could decide to share information about some web services and take the same decision. When a request from a community about the reputation of a web service arrives, the information services could agree on the answer to return. As a result, the requested information services would give the same information to the community. This scammed situation could be profitable for the lying information services in terms of rewards as they would receive higher payments.

Bibliography

[Asc11] Ascape guide. http://ascape.sourceforge.net/index.html, January 2011.

[BKG09] Jamal Bentahar, Babak Khosravifar, and Maziar Gomrokchi. Social network-based trust for agent-based services. In *AINA Workshops*, pages 298–303. IEEE Computer Society, 2009.

[CCMW10] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (wsdl) 1.1. http://www.w3.org/TR/wsdl, November 2010.

[Ecl11] Eclipse. http://www.eclipse.org/, January 2011.

[EMY+08] Said Elnaffar, Zakaria Maamar, Hamdi Yahyaoui, Jamal Bentahar, and Philippe Thiran. Reputation of communities of web services - preliminary investigation. In *AINA Workshops*, pages 1603–1608. IEEE Computer Society, 2008.

[GHM+10] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. Soap version 1.2 part 1: Messaging framework. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, November 2010.

[Gro04] Web Services Architecture Working Group. Web services architecture - w3c working group note. pages 6–9, 2004.

[Gro11] W3C Working Group. Qos for web services: Requirements and possible approaches. http://www.w3c.or.kr/kr-office/TR/2003/NOTE-ws-qos-20031125/, August 2011.

[Jav11] Java. http://www.java.com, January 2011.

[JFr11] Jfreechart. `http://www.jfree.org/jfreechart/`, January 2011.

[KBM+10] Babak Khosravifar, Jamal Bentahar, Ahmad Moazin, Zakaria Maamar, and Philippe Thiran. Analyzing communities vs. single agent-based web services: Trust perspectives. In *IEEE SCC*, pages 194–201. IEEE Computer Society, 2010.

[KBMT10] Babak Khosravifar, Jamal Bentahar, Ahmad Moazin, and Philippe Thiran. On the reputation of agent-based web services. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[KBT+09] Babak Khosravifar, Jamal Bentahar, Philippe Thiran, Ahmad Moazin, and Adrien Guiot. An approach to incentive-based reputation for communities of web services. In *ICWS*, pages 303–310. IEEE, 2009.

[MAS11] Mason. `http://cs.gmu.edu/~eclab/projects/mason/`, January 2011.

[MB07] Zaki Malik and Athman Bouguettaya. Evaluating rater credibility for reputation assessment of web services. In Boualem Benatallah, Fabio Casati, Dimitrios Georgakopoulos, Claudio Bartolini, Wasim Sadiq, and Claude Godart, editors, *WISE*, volume 4831 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2007.

[MS02] E. Michael Maximilien and Munindar P. Singh. Conceptual model of web service reputation. *SIGMOD Record*, 31(4):36–41, 2002.

[MS03] E. Michael Maximilien and Munindar P. Singh. An ontology for web service ratings and reputations. In Stephen Cranefield, Timothy W. Finin, Valentina A. M. Tamma, and Steven Willmott, editors, *OAS*, volume 73 of *CEUR Workshop Proceedings*, pages 25–30. CEUR-WS.org, 2003.

[OSB09] Martin J. OSBORNE. *An introduction to game theory - International edition*. Oxford University Press, 2009.

[udd10] Uddi 101. `http://uddi.xml.org/uddi-101`, November 2010.

[YMB+08] Hamdi Yahyaoui, Zakaria Maamar, Jamal Bentahar, Nabil Sahli, Said Elnaffar, and Philippe Thiran. On the reputation of communities of web services. In *Proceedings of the 8th international conference on New technologies in distributed systems*, NOTERE '08, pages 4:1–4:8, New York, NY, USA, 2008. ACM.

A

Simulation tool - technical presentation

In this chapter, we present in details the simulation tool we developed. We begin by presenting the choices we made for the modeling and the implementation.
An overview of the different entities follows.
The technical structure of the implementation is explained after that.
During a simulation, each entity of the environment acts according to a specific dynamic. The next section is composed of the implementations of each of these dynamics.
We end the presentation of our tool with a detailed review of every screen of the simulator.

Contents

A.1 Simulation choices

We introduce here the several choices we made regarding our simulation program. In the modeling section, we explain the simplifications that allowed us to develop a efficient implementation without overlooking key elements of our work. Elements that have been simplified are in majority out of the scope of the core of our investigation.

We then present the several implementation choices we made.

A.1.1 modeling

The first major choice we made was to stay the simplest possible and focus on what was really important, that is to say the interactions between web services, information services and communities. Thus, we intentionally overlooked some key elements and concepts that exists in the reality but were not really needed in the optic of our work.

The concept of user and the links he can have with web services and communities are not represented in our simulator because it would have introduced some unnecessary complexity. Consequently, the value of the reputation of a web service in our simulation is a static one. It does not vary along with the interactions the service has like it would in normal case. We decided here to assign a random value for each web service during its initialization.

Considering that we did not represent the interactions between information services and users, information services can not use feedbacks provided by the latter to compute the reputation value of each web service. A solution could have been to simply share the static value of the web services but, in this case, all the information services would have the exact same value for every web service. A situation that is not likely to happen in reality where it should exist some variations between the value that each information service possesses. We therefore decided to introduce a small random variation that is added to the actual reputation value and then transmit to the web services. This variation depends on one of the web service attributes, which is called “constancy”. The higher the constancy of a web service is, the lower is the chance for a information service to make correspond to this web service a reputation that is not the real one.

We also chose to categorise different web services in a dichotomic way. Depending on a threshold, a web service can be declared as good or bad. Since the interactions between several web services and users are not implemented, there is no way that a web service could improve its reputation level. Therefore, in order

to prevent that an important number of bad web services that would always be rejected by communities flooded the system, we imposed a defined limited number of attempts to join a community. Afterwards, web services are shut down.

Regarding the way we implemented the communities, we decided to get rid of all the master/slaves aspects. A master to dispatch them among the web services of the community was not necessary as there were no actual request from users to treat. Moreover, we proceeded in that way because we decided not to represent the attraction/retention capabilities that a master should normally have. In our simulator, the joining requests always come from web services that want to enter a community. And when a web service joins a community, we considered that he never leaves it. Therefore, we chose to limit our implementation of a community to a list of different web services. The algorithm responsible for the acceptation or the rejection of a joining request is handled inside the community but not delegated to a specific web service.

Communities do not always need to add new web services. In some cases, their performances are such that they do not want to let new web services join them. In our simulation, we do not consider that fact and assume that communities remain open about the idea of accepting new web services and do not have a limit regarding the number of web services they let join.

In reality, mechanisms like UDDI registries help the different entities to search for the others. For example, a web service uses these registries to discover the communities that exist in the system. Here, in order to simplify and avoid the management of such registries, we decided to implement a unique object Repository that references all the web services, communities and information services. Moreover, we used the singleton pattern so that all the different entities can easily call that repository.

We decided to let a specific object handle the data that serves as results. We also implemented it using the singleton pattern. The concerned entities can therefore easily call the object and transmit some information to it.

A.1.2 Implementation choices

We decided to develop our solution with the Java programming language in its 1.6 revision [Jav11]. As SDK, we used Eclipse 3.6 (Helios) [Ecl11]. In order to display the results of the simulation with graphs, we added to our project the JFreeChart library [JFr11].

We also wanted our simulation program to be user-friendly. Thus, we decided to develop a simple user interface. The latter basically contains two screens. The first allows the user to configure the upcoming simulation and the second shows the obtained results in real time. We developed our interface according to the Model-View-Controller (MVC) pattern. In other words, our code is divided so that some classes are designed to handle the data, some to present this information to the user of the program and some are intended to control the interface.

The Observer pattern is also used in order to update the interface with the data transmitted by the several entities.

A.2 Entities overview

In this section, we review each main entity in order to analyse their goals in the system, the hypothesis we made about them for the simulation and the parameters that compose them.

A.2.1 Web service

Parameters

- **Id**
  This number identifies the web service among all the others.
- **Quality of service**
  This value expresses the quality of service the web service can offer when answering to a customer's request.
- **Quality of service variability**
  This value expresses the percentage of variability of the web service's quality of service. This value has been introduced in the program to simulate the fact that web services can not always answer to requests with the same quality of service (if overloaded for example). The bigger is this value, the bigger is the chance for the web service, if asked, to return a QoS value different from the real one. However, the difference between the returned QoS and the real one does not vary too much.

- **Bribe value**
  If a web service wants to pay an information service in order to motivate this information service to lie about its reputation, this value represents the value of the initial reward the web service promises to the information service for this lie.

- **Bribe step**
  If the initial bribe value is not enough to motivate the information service to lie, the bribe step value is the one by which the web service increases its bribe.

- **Number maximum of attempts**
  If the web service is either not accepted or rejected by a community, this number represents the number of times the web service can try to enter another community.

Goals

- The only goal of a web service is to be accepted in a community.

Hypothesis

We assume that :

- No web service wants to work on its own but all want to join a community.

- Web services do not have expectations regarding the communities they want to enter. As a result, the choice of the community to request is made randomly.

- A web service always wants an information service to lie about its reputation if the community the web service wants to enter has a too big minimum accepted reputation value.

- Web services do not change their quality of service over time.

- As a consequence to the precedent point, communities will not accept a web service they already rejected, and a a web service will not try to enter a community it has already been rejected from.

- A web service only joins one community at a time.

- A web service tries to join one community. If the web service manages to be accepted and to stay in a community, it will never want to leave this community.

A.2.2 Community of web services

Parameters

- **Id**
  This number identifies the community of web services among all the others.
- **Number of used information services**
  When a community wants to have information about a web service wanting to join it, the community asks to different information services about this web service. This value indicates the number of different information services the community queries.
- **List of known information services**
  This list registers all the information services a community knows. When the community asks for information about a web service, the community selects information services from this list to ask them.
- **List of web services in the community**
  This list registers all the web services that are in the community.
- **web service minimum accepted reputation**
  This value indicates the minimum reputation for a web service to have in order to join the community.

Goals

- The goal of a community of web services is to only accept and keep *good* web services (the concept of *good* web service is explained in the next point).

Hypothesis

We assume that :

- A community considers a web service as a good one if the reputation of this web service is at least equal to the minimum accepted reputation value of the community.
- Communities do not have a maximum capacity. As a result, they always accept web services as long as the reputation of those web services are good enough.
- A community does not know all the information services in the system. The only information services the community knows is a set of randomly chosen information services. The size of this set is, if possible (if there are enough information services in the system) equal to 2*`number_of_used_information_services`. If there are not enough information services to equal this number, the community simply knows all the information services in the system.

- Each time a community wants to get information about a web service, the community asks about it to a group of information services. This group is composed of
`number_of_used_information_services` information services. Those information services are the
`number_of_used_information_services - 1` information services that have the best trust value according to the community and the last one is the information service that has the worst trust value according to the community. The latter is chosen in order to give a chance to the "worst" information services to improve their trust value.

A.2.3 Information service

Parameters

- **Id**
This number identifies the information service among all the others.

- **Truth trend value**
This value indicates the probability for an information service to tell the truth to a community of web services if the information service has the choice to lie.

Goals

- The goal of an information service is to increase the rewards it can get from the communities on one hand, and from the web services on the other hand.

Hypothesis

We assume that :

- An information service which has good truth value according to communities of web services will be often requested to give information. On the contrary, an information service which is not often requested is more likely to have poor truth value according to the communities. Those, more than the others, need to increase their trust value towards the communities. Therefore, each time an information service is not requested for a specific amount of time (easily modifiable in the code), its
`truth_trend_value` increases.

A.3 Architecture

In this section, we review the architecture we adopted for our program. We take a look at each class we coded and briefly explained its purpose.

Our main Java project is composed of the six following packages :

- **controller :**
  The controller package contains a single class Controller. This Controller serves as entry point for the entire application. Indeed, the Controller is in charge of initializing the interface and the different entities necessary for the simulation. The Controller is also responsible for handling the events triggered by the interface, for example when the user presses a button.

- **entities :**
  This package contains the implementation of the different entities that are used for the simulation such as web services or information services. We detail later each of these entities.

- **exceptions :**
  As its name suggests, the exceptions package contains all the exceptions that can be thrown by the application during its execution.

- **gameTheory :**
  The gameTheory package essentially contains classes related to the game theory mechanisms. More explanations on these classes can be found later in the document.

- **observer :**
  In order to make it more suitable to our needs, we redefined the Observer pattern with the methods that were useful to us. In this package, we can thus find our version of the two interfaces Observer and Observable.

- **utils :**
  The utils package contains some classes that are used as tools during the simulations. The Rule class represents a rule to compute the payments of trust update. The Stats class is used to gather information about the current simulation. Those data can then be transmitted to the view in order to provide results to the user. Finally, the WebServicesStarter class allows us to prevent all the web services set for the simulation to start simultaneously. The Stats and the WebServicesStarter have been implemented using

the singleton pattern.

- **view :**
  The view package contains all the classes used to display the interface of the simulation program.

We now examine in more details the content of the entities and the gameTheory packages.

The entities package

As we said, the entities package contains our implementation of the entities whose behaviour is observed through the running simulations. Therefore, we find here the WebService, InformationService and Community classes. We also put in this package the Repository, isGroup and Report classes.

- **Community :**
  As we explained in Section A.1.1, we didn't implement the concepts of master/slaves in this implementation of the community. Therefore, a community contains simply a list of all the web services that are inside but without introducing a special distinction between them. A community is identified by an ID number and also possesses a list that maps to each information service that the community has had contact with a trust value. A special value called acceptanceReputation indicates the minimum reputation level that the community requests to accept a new web service. The Community class implements the Runnable interface in order to run in its own thread.

- **InformationService :**
  The InformationService class is not a very complicated one. Each information service maintains a list that maps web services with a reputation value. An ID number identifies each information service. This class also implements the Runnable interface so that each information service runs within a thread.

  The behaviour of a information service is here fairly simple. It reacts to the requests that come from communities and shares with them the information it knows about web services.

- **IsGroup :**
  When a community searches information about a particular web service, it requests a group of information services. The IsGroup class represents this group.
  The IsGroup is in charge of the game theory part of the simulator. When information is requested for a community, each information service from the created IsGroup uses game theory in order to decide which decision it should make and which information it should provide.

- **Report :**
  The Report class represents the list of reputation values that a community received from the information services it requested about the quality of a particular web service.

- **Repository :**
  To avoid the complexity of dealing with multiple registries for the localisation of the entities, we decided to implement a single component called Repository. This component knows all the entities that are present in the system. Thus, it maintains a list of communities, a list of information services and a list of web services.

  We developed the Repository class following the singleton pattern. All the communities, information services and web services can thus access this Repository and send it requests.

- **WebService :**
  The WebService contains a number of attributes that define the web service. First, an ID number is used to identify the service. A float value indicates the reputation value of the web service and a boolean detects if the service belongs to a community or not.

  As we explained in the section dedicated to the choices of implementation, we introduced some randomness when it comes to the knowledge that information services hold about the reputation of the web services. This has been implemented through a method that returns alternated version of the reputation of the requested web service.

  As for Community and InformationService, the WebService class implements the Runnable interface that allows each instance of he class to run

within its own thread. The latter stops when the web service enters a community or reaches the maximum number of attempts.

The gameTheory package

The gameTheory package contains a group of classes that implement concepts related to the game theory. We find here the classes that represent the ActionProfile, the Choices, the Outcome and the Game. The way we implemented these classes restricts the game to a two-players play.

- **ActionProfile :**
  An ActionProfile corresponds to the couple formed by the Choices made by the players and the Outcome that the players gains in the current situation.

- **Choices :**
  This class represents the choices made by each of the two players. These choices are coded using integers.

- **Game :**
  The Game class is the most complicated of this package. It contains a list that maps choices to outcomes that model the game. Two integer values indicate the number of actions that the two players can choose.

  When a game has been initialised, we can apply some methods on it in order to compute results. For example, a method allows us to find the best response function for a given player. Another useful function helps us to find the Nash equilibrium of the game.

- **Outcome :**
  The Outcome corresponds to the gains that each player can receive according to a certain situation.

A.4 Simulation dynamic

In this section, we explain what happens during a simulation for each entity.

A.4.1 Web service

1. All web services are created according to the `population` and `reputation` data chosen by the user in the configuration screen.

2. Web services are put in a list and wait to enter the system by groups. This is a technical choice we made in order not to overload the computer running the simulation.

3. When a web service enters the system, it asks a community randomly chosen to join it.

   (a) The community does not accept the web service, the web service goes back to step 3, as long as its `maximum_number_of_attempts` is not reached.

   (b) The community accepts the web service but tests it and rejects it after a time, the web service goes back to step 3, as long as its `maximum_number_of_attempts` is not reached.

   (c) The community accepts the web service and keeps it.

4. The web service becomes inactive and another one can enter the system.

A.4.2 Community of web services

1. All web services are created according to `used_information_services` and `web_service_minimum_accepted_reputation` data chosen by the user in the configuration screen.

2. The community creates a sorted list of the known information services. In this list, the information services are ordered by the trust the community has towards them.

3. At regular intervals, the community tests its web services (simulating the mechanism of sending requests).

   (a) If a web service passes the test, it stays in the community.

   (b) If a web service does not pass the test, the community ejects the web service.

4. Each time a web service asks the community to join, the community creates a group of information services and asks them about the reputation of the web service. This group is composed of the `used_information_services - 1` first information services of the community list (created at step 2) and of the last one of this list.

5. The community gets acquainted with the average reputation.

   (a) If this value is good enough (equals or is superior to the community `web_service_minimum_accepted_reputation`), the community accepts the web service.

   (b) If this value is not good enough (inferior to the community `web_service_minimum_accepted_reputation`), the community does not accept the web service.

A.4.3 Information service

1. All information services are created according to the `truth_trend` data chosen by the user in the configuration screen.

2. The information service gets acquainted with the reputation of all the web services in the system.

3. The information service waits for a specific amount of time (called `session`) to be requested by a community to give information about a web service reputation.

   (a) If the information service is requested by a community, it analyses (thanks to game theory) the best action to make (tell the truth or lie to the community about the web service reputation) and informs the community of the consequent reputation.

   (b) If the information service is not requested by a community within this time, the `truth_trend_value` of the information service increases.

A.5 Screens overview

In order to make the usage of the simulator more user-friendly, we decided to implement a user interface. The latter should help the user to prepare and configure simulations in an easy way. The application consists essentially in two main activities. In the first, the user configures the upcoming simulation while in the second, the results of the simulation are displayed in real-time.

A.5.1 Configuration - Entities

This first configuration screen allows the user to configure the entities.

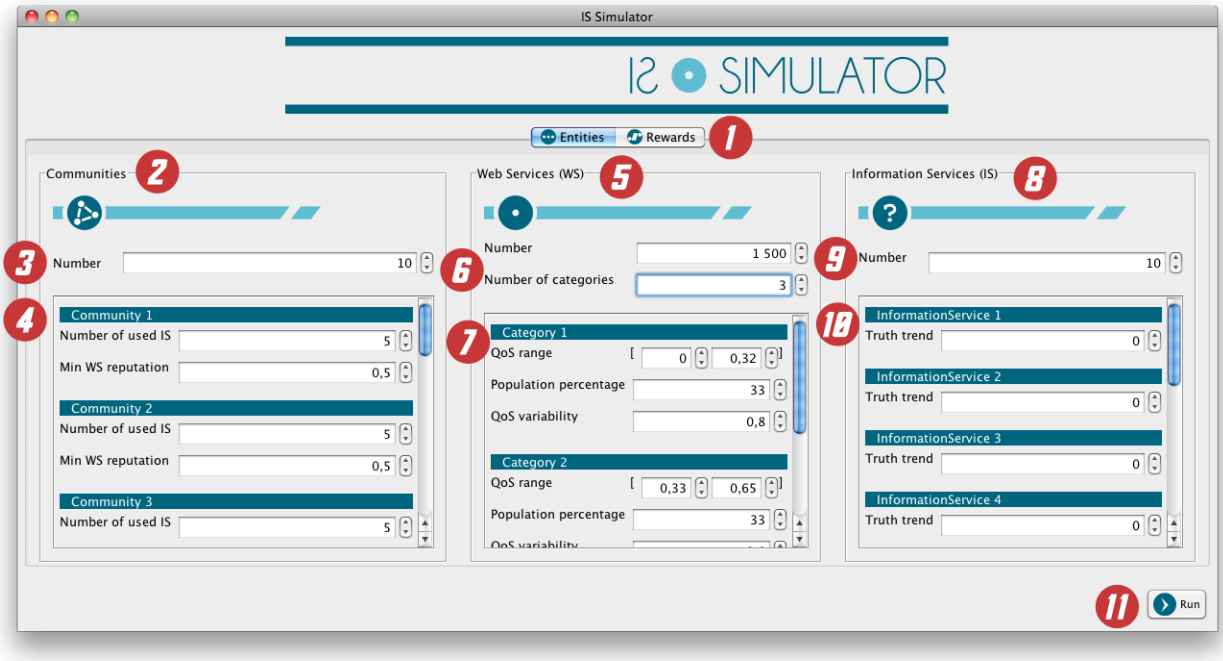

Figure A.1: Configuration screen - Entities

In **(1)**, the user can select one of the two configuration screen. "Entities" corresponds to the screen allowing to modify the several parameters regarding the agents while "Rewards" lets the user set the value of the rewards, incentive, β function, γ function and τ function.

In the panel identified by **(2)**, the user can configure the communities. The number of communities that will be deployed during the upcoming simulation

can be selected using **(3)**. In the panel identified by **(4)**, the parameters of each community can be set :

- **Number of used IS :** Set the number of information services used by the community when requesting information about a web service.

- **Min WS reputation :** Set the minimum reputation value required by the community to allow a web service to join.

In the panel identified by **(5)**, the user can configure the web services. The number of web services that will be deployed during the upcoming simulation and the number of categories in which they will be distributed can be selected using **(6)**. In the panel identified by **(7)**, the parameters of each category of web services can be set :

- **QoS range :** Set the range in which the QoS of the Web services of the category will vary.

- **Population percentage :** Set the percentage of the total web services that will belong to the category.

- **QoS variability :** Set the QoS variability of the web service.

In the panel identified by **(8)**, the user can configure the information services. The number of information services that will be deployed during the upcoming simulation can be selected using **(9)**. In the panel identified by **(10)**, the truth trend parameter of each information service can be set.

By clicking the button identified by **(11)**, the user starts the simulation with the parameters that have been chosen.

A.5.2 Configuration - Rewards

The second section of the configuration screen is used to set parameters regarding the rewards and incentive.

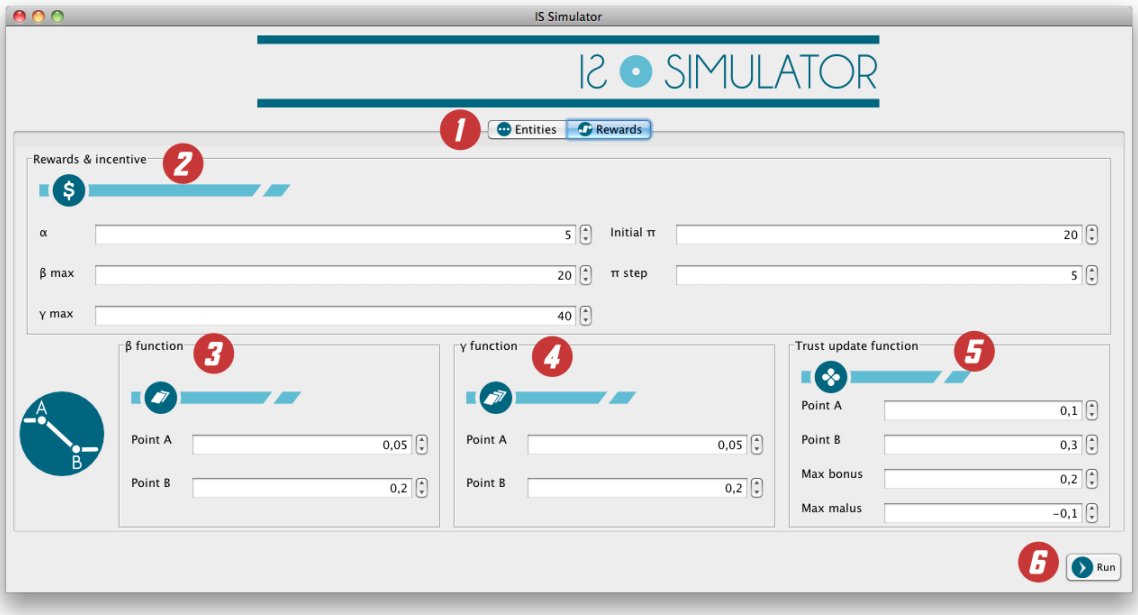

Figure A.2: Configuration screen - Rewards

In **(1)**, the user can select one of the two configuration screen. "Entities" corresponds to the screen allowing to modify the several parameters regarding the agents while "Rewards" lets the user set the value of the rewards, incentive, β function, γ function and τ function.

In the panel identified by **(2)**, the user can modify the value of the rewards α, β max and γ max and the parameters related to the incentives, initial π and π step.

In the panel identified by **(3)**, the parameters of the β function can be adjusted. This function has been defined in Section 4.4 in Definition 13. The shape of this function has been presented in Figure 4.2 and the Point A parameter represents the β_a while Point B represents β_b.

In the panel identified by **(4)**, the parameters of the γ function can be adjusted. This function has been defined in Section 4.4 in Definition 15. The shape

of this function has been presented in Figure 4.3 and the Point A parameter represents the γ_a while Point B represents γ_b.

In the panel identified by **(5)**, the parameters of the τ function can be adjusted. This function has been defined in Section 4.5 in Definition 16. The shape of this function has been presented in Figure 4.4 and the Point A parameter represents the τ_a while Point B represents τ_b. The user can also set the value of the maximum bonus and the maximum malus.

By clicking the button identified by **(6)**, the user starts the simulation with the parameters that have been chosen.

A.5.3 Results - QoS repartition

This screen displays the repartition of the web services of the current simulation according to their quality of service value.

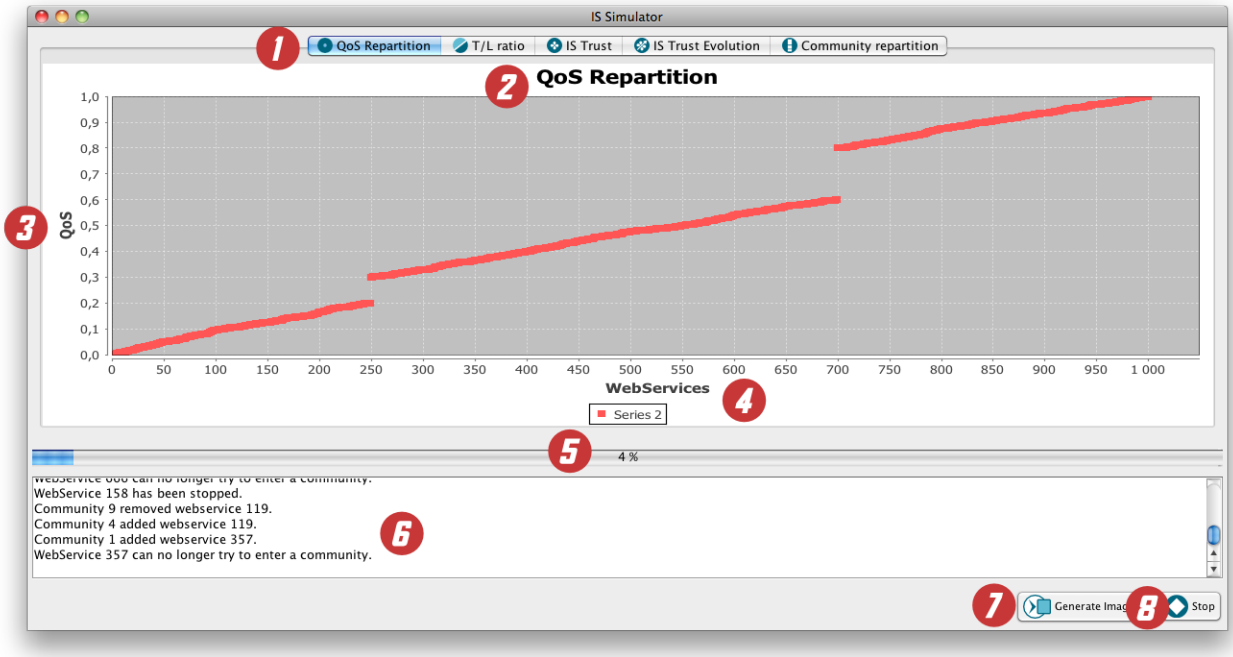

Figure A.3: Results screen - QoS repartition

In **(1)**, the user can select one of the five results screens.

In **(2)**, a chart that represents the repartition of the web services according to their quality of service is displayed. The Y-axis **(3)** represents the quality of service of a web service while the X-axis **(4)** represents all the web services deployed during the current simulation.

The progress bar identified by **(5)** indicates the progression of the current simulation.

In the area identified by **(6)**, real-time notifications about the current simulation are displayed.

By clicking the button identified by **(7)**, PNG image files of the results charts are generated and saved on the computer.

By clicking the button identified by **(8)**, the user can interrupt the current simulation.

A.5.4 Results - Truth/Lie ratio

This screen displays the evolution of the Truth/Lie ratio. This ratio corresponds to the total amount of times the information services chose to tell the truth compared to the total of choices they had to make.

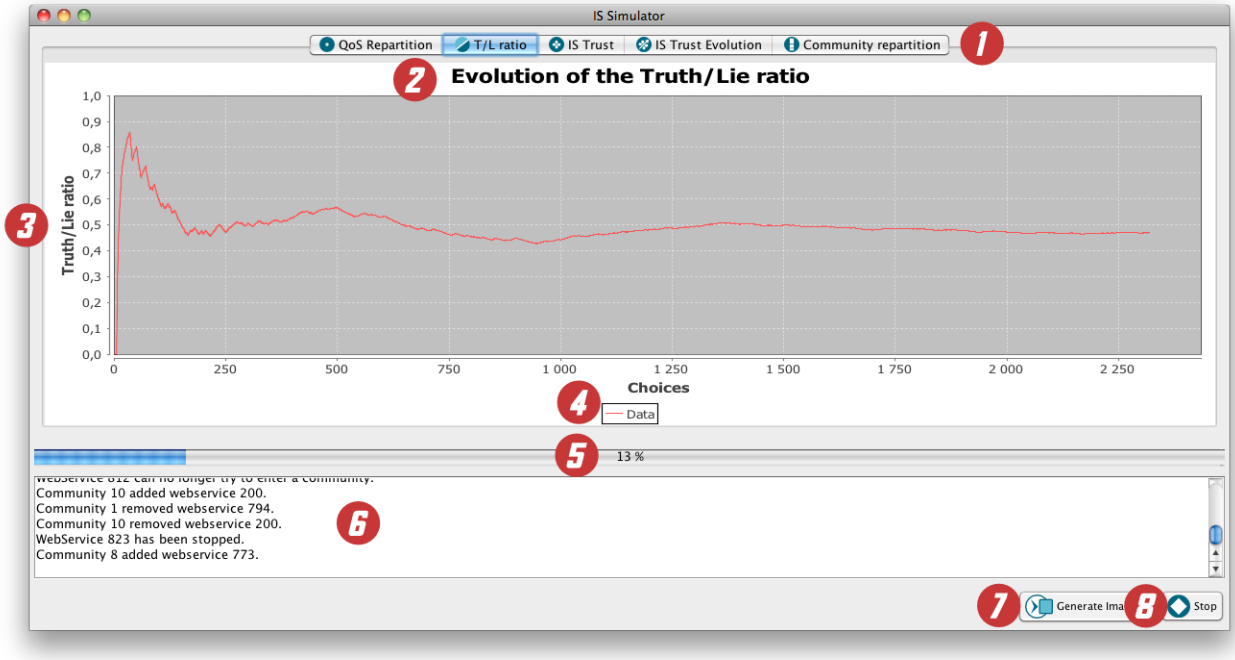

Figure A.4: Results screen - Evolution of the Truth/Lie ratio

In **(1)**, the user can select one of the five results screens.

In **(2)**, a chart that represents the evolution of the Truth/Lie ratio is displayed. The Y-axis **(3)** represents the Truth/Lie ratio. Each time a community asks information services for information about a web service, each of these information service can choose either to lie or tell the truth. The X-axis **(4)** represents each of these choices.

The progress bar identified by **(5)** indicates the progression of the current simulation.

In the area identified by **(6)**, real-time notifications about the current simulation are displayed.

By clicking the button identified by **(7)**, PNG image files of the results charts are generated and saved on the computer.

By clicking the button identified by **(8)**, the user can interrupt the current simulation.

A.5.5 Results - Trust average

This screen displays the trust average and the truth trend of each information service.

Figure A.5: Results screen - Trust average and truth trend

In **(1)**, the user can select one of the five results screens.

In **(2)**, a chart that represents the current trust average and truth trend of each information service deployed in the running simulation is displayed. The trust value that a particular community has towards an information service evolves according to the τ function we defined in Section 4.5. In this chart, we show the average trust that all the communities have towards the deployed information services. The Y-axis **(3)** represents the value of the trust average and the truth trend. The trust average can vary between -1 and 1 while the truth trend can vary between 0 and 1. The X-axis **(4)** indicates all the information services that have been deployed in the current simulation. The colours used to distinguish the truth trend and the trust average are shown in **(5)**.

The progress bar identified by **(6)** indicates the progression of the current simulation.

In the area identified by **(7)**, real-time notifications about the current simulation are displayed.

By clicking the button identified by **(8)**, PNG image files of the results charts are generated and saved on the computer.

By clicking the button identified by **(9)**, the user can interrupt the current simulation.

A.5.6 Results - Evolution of the trust average

This screen displays the evolution of the trust average of each information service.

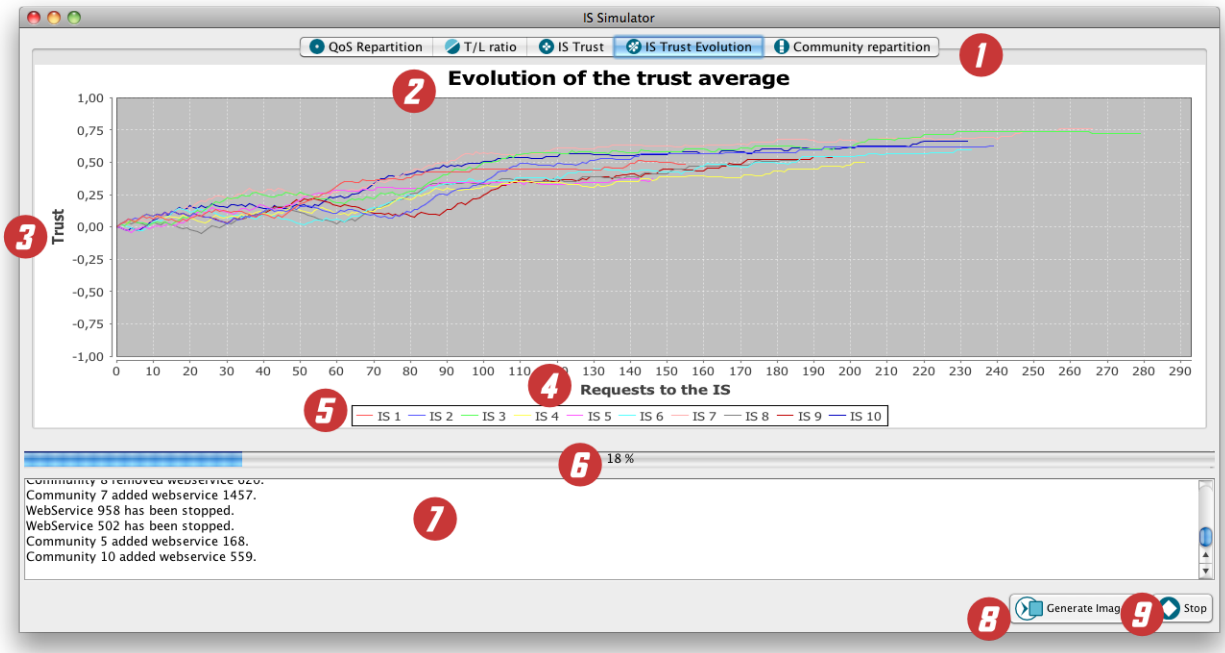

Figure A.6: Results screen - Evolution of the trust average

In **(1)**, the user can select one of the five results screens.

In **(2)**, a chart that represents the evolution of the trust average of each information service deployed in the running simulation is displayed. The trust value that a particular community has towards an information service evolves according to the τ function we defined in Section 4.5. In this chart, we show the evolution of the average trust that all the communities have towards the deployed information services. The Y-axis **(3)**) represents the value of the trust average. The trust average evolves each time an information service is requested information about a web services by the communities. The X-axis **(4)** indicates all of these requests. The colours used to distinguish the trust average of each information service are shown in **(5)**.

The progress bar identified by **(6)** indicates the progression of the current simulation.

In the area identified by **(7)**, real-time notifications about the current simulation are displayed.

By clicking the button identified by **(8)**, PNG image files of the results charts are generated and saved on the computer.

By clicking the button identified by **(9)**, the user can interrupt the current simulation.

A.5.7 Results - Community repartition

This screen displays the repartition of the web services of each community.

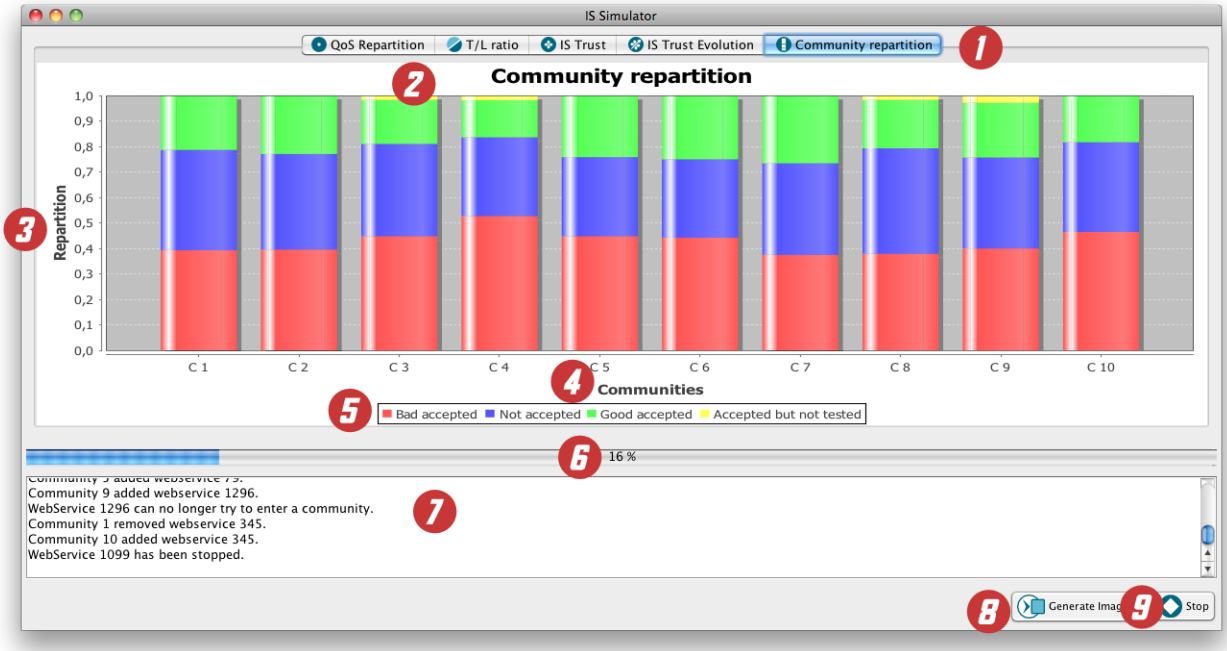

Figure A.7: Results screen - Community repartition

In **(1)**, the user can select one of the five results screens.

In **(2)**, a chart that represents the current repartition of the web services in each community deployed in the running simulation is displayed. There are four different types of web services : bad web services that have been accepted, bad web services that have been rejected, good web services that have been accepted and web services that have been accepted but not tested yet (in other words, the community does not know if the web service is really good or bad). The Y-axis **(3)** represents the repartition of each type of web services while the X-axis **(4)** indicates all of the communities that have been deployed in the current simulation. The colours used to distinguish the different types of web services are shown in **(5)**.

The progress bar identified by **(6)** indicates the progression of the current simulation.

In the area identified by **(7)**, real-time notifications about the current simulation are displayed.

By clicking the button identified by **(8)**, PNG image files of the results charts are generated and saved on the computer.

By clicking the button identified by **(9)**, the user can interrupt the current simulation.